OPTIMUM STRATIFICATION

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

Master of Philosophy

IN

STATISITCS

By

Mrs. Neeru Rani

Under the Supervision of

Prof. M.J. Ahsan

DEPARTMENT OF STATISTICS AND OPERATIONS RESEARCH
ALIGARH MUSLIM UNIVERSITY
ALIGARH (INDIA)

2008

Dedicated

To

My

Parents

and Husband

+91-571-2701251 (O)
+91-571-2404221 (R)

DEPARTMENT OF STATISTICS AND OPERATIONS RESEARCH

ALIGARH MUSLIM UNIVERSITY, ALIGARH-202 002

Prof. M. Jameel Ahsan
M Sc , M. Phil., Ph.D. (Alig)

Dated 15.09.2008

Certificate

This is to certify that **Mrs. Neeru Rani** has carried out the work reported in this dissertation entitled "**OPTIMUM STRATIFICATION**" under my supervision. This dissertation is her own work and I recommend it for consideration for the award of **Master of Philosophy in Statistics.**

(Prof. M. J. AHSAN)
Supervisor

4/663 (D), Shabistan Compound, Dodhpur, Aligarh - 202 001 (U.P.)

ACKNOWLEDGEMENT

I feel great pleasure at the completion of this work and would like to show deep indebtedness and sincere gratitude to all who helped me in the completion of this work.

I owe a deep sense of gratitude to my supervisor, **Prof.M. J. AHSAN,** Department of Statistics & Operations Research, A.M.U., Aligarh for not only being dedicated and perseverant but also for his guidance, patience, great involvement and sympathetic behaviour, which enabled me to complete this work within the stipulated time.

I am thankful to **Prof. Abdul Hameed Khan,** Chairman, Department of Statistics and Operation Research, Aligarh Muslim University, Aligarh, for providing me necessary facilities in the department and for extending his kind cooperation at every stage of the writing of this dissertation.

I would like to express thanks to all my teachers Prof. M. Z. Khan, Dean Faculty of Science, Prof. M. M. Khalid, Dr. H. M. Islam, Dr. Mohd. Yaqub, Dr. A. Bari , Mr. S. S.Hasan, Dr. S. M. Arshad, Dr. R. U. Khan, Dr. Shakeel Javaid, Dr. Kamal-Ullah, Dr. Haseeb Ather and Dr. (Mrs.) Najmussehar for their concern and counsel.

My sincere thanks are also due to all non-teaching staff members of the department for their kind help and cooperation.

I am greatly indebted to my research colleagues and friends who have encouraged me throughout this work. I am particularly thankful to Ms. Yojna, Mr. Rahul Varshney, Mr. Devendra Kumar,

Ms. Shazia Zarrin, Ms. Shazia Ghufran, Ms. Samman Khowaja, Mrs. Trapti Thakur, Ms. Shashi Saxena, for their cooperation and sharing of ideas during the preparation of this manuscript.

My heart goes out in reverence to my parents and brother for their tremendous patience, forbearance, endurance and affection. All my appreciation for their support will not be adequate and enough to match their good wishes.

This acknowledgement will remain incomplete without giving thanks to my husband without whose support, encouragement cooperation, involvement and sympathetic behavior it would not be possible for me to complete this work.

Date:15.09.2008

Neeru Rani

Neeru Rani
Research Scholar
Department of Statistics and
Operations Research

CONTENTS

PREFACE

This dissertation entitled **"Optimum Stratification"** is submitted to the Aligarh Muslim University, Aligarh, India, for the partial fulfilment of the degree of Master of Philosophy in Statistics. It embodies literature survey work carried out by me in the Department of Statistics and Operations Research, Aligarh Muslim University, Aligarh, India.

In this dissertation the problem of stratification is studied under different situations. Apart from the classical approaches some recent works using mathematical programming to solve the problem of stratification are also disscussed.

This dissertation consists of four chapters.

Chapter 1 deals with the basic ideas of sampling theory and Mathematical Programming. It describes the basic concepts and results, that are relevant to the later chapters.

Chapter 2 deals with the problem of optimum stratification. For stratified sampling to be efficient the strata should be internally homogenous as far as possible, with respect to the study variable. In order to achieve this goal the stratum boundaries are so chosen that the stratum variances are

as small as possible. Keeping this in mind many researchers have tackled this problem. The concerned literature is surveyed and some results have been discussed in detail in this chapter.

Chapter 3 deals with the problem of stratification as a mathematical programming problem. The problem of finding the Optimum Strata Boundaries (OSB) has been formulated as a Mathematical programming problem (MPP) such that the variance of the estimated population parameter is minimum under Neyman allocation subject to the constraint that the sum of the widths of the strata is equal to the range of the stratification variable. The formulated MPP is then considered as a multistage decision problem which could be solved by Dynamic Programming Technique.

Chapter 4 deals with the application of dynamic programming technique when the stratification variable follows some specialized distributions. Optimum Strata Boundaries (OSB) are obtained through Optimum Strata Widths (OSW) for Rectangular, Right-Triangular, Exponential and Log-Normal distributions of stratification variables.

A comprehensive list of references, arranged in alphabetical order is also provided at the end of this dissertation.

CHAPTER -I

INTRODUCTION

1.1 SAMPLING:

Sampling is the manner or scheme through which the required numbers of units are selected in a sample from a population. The purpose of sampling is to as collect maximum information about the population under consideration at minimum cost, time and man power. Sampling is also the science and art of controlling and measuring reliability of useful statistical information through the theory of probability.

1.2 CENSUS:

The process of obtaining information about a population by enumerating each and every unit of the population is called census or complete enumeration. Usually census is carried out to collect information about births, deaths, occupations, social and economic conditions of the people of the country at a given point of time. Almost in all the world population census is conducted at regular intervals of time, usually ten years.

1.3 CENSUS VERSUS SAMPLING:

In census, each and every unit of the population is studied but in sampling only a selected number of units are studied. The results in census are based on all units whereas, in sampling the results are based on the data of these units (selected numbers) are supposed to yield information about the whole population. The cost of covering all units would be greater than that of covering only a sample fraction, so the sample surveys will usually be less costly than census.

The results from a carefully planned and well executed sample surveys are expected to be more accurate than those from a complete census.

In general we have the following advantages of sampling over census.

1- Reduced cost

2- Reduced time

3- Administrative convenience

4- Greater speed

5- Greater scopes

6- Sampling is more scientific because it is based on sound statistical theory.

Apart from the above advantages sampling has some disadvantages also as given below:

1. If the sample surveys are not carefully planned and executed, inaccurate and misleading results can appear.
2. Sample surveys need highly trained peronells and sophisticated instruments. In absence of these facilities, the results may not be reliable.
3. A sample must be a true representative of the population. If the sample is not properly selected or the sample size is inadequate it may fail to show the characteristics of the population.
4. A sample survey can not provide information about individual units.
5. All sampling results are subjected to some error, called sampling error. However, using statistical theory we can have an estimate of this error.

1.4 SIMPLE RANDOM SAMPLING WITH REPLACEMENT:

A simple sample is drawn unit by unit. In simple random sampling with replacement (SRSWR) the selected unit is replaced in the population before the next draw, so that all the population units are available for selection at every draw. In SRSWR the same unit of the population may be included more than once in the sample. There are N^n possible samples of size n outs of a population of size N and each of the N^n samples have an the equal probability $\frac{1}{N^n}$ of being selected.

1.5 SIMPLE RANDOM SAMPLING WITHOUT REPLACEMENT:

Suppose the population consist of N units, then in simple random sampling without replacement (SRSWOR) the units are drawn one by one in such a way that a unit drawn at a time is not returned to the population before the next draw is made. This process is repeated n times. The same unit of the population cannot occur more than once in the sample. The possible number of distinct samples of a fixed size n is ${}_{N}C_{n}$ and each sample has an equal probability $\frac{1}{{}_{N}C_{n}}$ of being selected.

1.6 STRATIFIED SAMPLING:

STRATIFIED RANDOM SAMPLING:

The precision of an estimator of the population parameters depends on the size of the sample and the variability or heterogeneity among the units of the population. If the population is very heterogeneous and considerations of the cost limit the size of the sample, it may be found impossible to get a sufficiently precise estimate by taking a simple random sample from the entire population. The solution of this problem lies in Stratified Sampling Design. In Stratified Sampling the population of size N is divided into L non-overlapping and exhaustive groups called Strata. Each of which is relatively more homogeneous as compared to the population as a whole.

Independent simple random samples of predetermined sizes from each stratum are drawn and the required estimators of the population parameters are constructed.

1.6.1 PRINCIPAL REASONS FOR STRATIFICATION:

1) To gain in precision, we may divide a heterogeneous population into strata in such a way that each stratum is internally as homogeneous as possible.

2) For administrative convenience in organizing and supervising the field work. Stratified sampling is best suited.

3) To obtain separate estimates for some part of the population.

4) We can accommodate different sampling plans in different strata.

5) We can have data of known precision for certain sub divisions, consisting of one or more strata and each sub division is treated as a separate population.

6) Sampling problems may differ markedly in different parts of the population. With the human populations, people living in institutions like hotels, hospitals etc. are often placed in a different stratum from people living in ordinary homes because a different approach to the sampling is appropriate for the two situations.

1.6.2 SOME RESULTS IN STRATIFIED SAMPLING:

NOTATIONS:

Let N, be the population size.

n, be the sample size.

In h^{th} stratum

let N_h, be the stratum size.

n_h, be the sample size.

$f_h = \frac{n_h}{N_h}$, be the sampling fraction.

$W_h = \frac{N_h}{N}$, be the stratum weight.

y_{hj}; $j = 1,2,\cdots,N_h$, be the value of the characteristic under study for the j^{th} unit.

$\bar{Y}_h = \frac{1}{N_h}\sum_{j=1}^{N_h} y_{hj}$, be the stratum mean.

$\bar{y}_h = \frac{1}{n_h}\sum_{j=1}^{n_h} y_{hj}$, be the sample mean.

$S_h^2 = \frac{1}{N_h - 1}\sum_{j=1}^{N_h}(y_{hj} - \bar{Y}_h)^2$, be the variance in the stratum.

$s_h^2 = \frac{1}{n_h - 1}\sum_{j=1}^{n_h}(y_{hj} - \bar{y}_h)^2$, be the sample variance.

Further let

$Y = \sum_{h=1}^{L}\sum_{j=1}^{N_h} y_{hj} = \sum_{h=1}^{L} N_h \bar{Y}_h$, be the population total.

and $\bar{Y} = \frac{1}{N}\sum_{h=1}^{L}\sum_{j=1}^{N_h} y_{hj} = \frac{Y}{N} = \frac{1}{N}\sum_{h=1}^{L} N_h \bar{Y}_h = \sum_{h=1}^{L} W_h \bar{Y}_h$, be the population mean.

Let the aim of the sample survey is to estimate the population mean $\bar{Y}$.

Define $\bar{y}_{st}$, the stratified sample mean as

$$\bar{y}_{st} = \sum_{h=1}^{L} W_h \bar{y}_h$$

Since the sampling is simple random sampling within each stratum, $\bar{y}_h$ is an unbiased estimator of $\bar{Y}_h$ because

$$E(\bar{y}_{st}) = \sum_{h=1}^{L} W_h E(\bar{y}_h) = \sum_{h=1}^{L} W_h \bar{Y}_h = \bar{Y}$$

Also $V(\bar{y}_{st}) = V(\sum_{h=1}^{L} W_h \bar{y}_h)$

$$= \sum_{h=1}^{L} W_h^2 V(\bar{y}_h)$$

$$= \sum_{h=1}^{L} W_h^2 \left(\frac{1}{n_h} - \frac{1}{N_h} \right) S_h^2 \qquad \text{(using result of SRS)}$$

$$= \sum_{h=1}^{L} \left(\frac{1}{n_h} - \frac{1}{N_h} \right) W_h^2 S_h^2$$

$$= \sum_{h=1}^{L} \left(\frac{N_h - n_h}{N_h} \right) \frac{W_h^2 S_h^2}{n_h}$$

$$= \sum_{h=1}^{L} (1 - f) \frac{W_h^2 S_h^2}{n_h}$$

The three main problems arising in using the stratified sampling are-

(1) Determining the optimum number of strata

(2) Determining the optimum stratum boundaries

(3) Determining the optimum size of the sample to be selected from every stratum (the allocation problem)

Before constructing the strata the sampler has to decide about the numbers of strata (L) considering the following points.

(i) At the rate does the variance of the estimate decreases as L increases?

(ii) How the cost of the survey is affected by an increase in the number of strata.

Usually an auxiliary variable which is highly correlated with the main study variable is used to estimate the number of strata. Cochran (1977) showed

that if the correlation between the main and auxiliary variables is less than .95 little reduction in the variance of the estimate is expected for $L > 6$.

1.6.3 THE PROBLEM OF STRATIFICATION:

Once the number of strata L is fixed one needs the $(L-1)$ cut off points of the distribution of the main variable x itself to construct L strata. In this dissertation the problem of determining the strata boundaries is discussed in detail. When a single characteristic is under study and its frequency distribution is known it can be used for determining the strata boundaries. Dalenius and Gurney (1951), Mahalanobis (1952), Hansen Hurwitz and Madow(1953), Aoyama(1954),Dalenius and Hodges(1959), Durbin(1959), Ekman(1959), Sethi(1963), Murthy(1967) and several other authors used the frequency distribution of the main study variable x for determining the strata boundaries under various allocation of the sample sizes.

Dalenius (1957) worked out the best approximate stratum boundaries under Neyman Allocation.

1.6.4 THE ALLOCATION PROBLEM:

After fixing the number of strata and their boundaries the sampler has to decide about the size of the sample to be drawn from each stratum. This problem is known as "The problem of allocation" in sampling literature. The

sample sizes may be worked out to minimize the variance of the estimate for a fixed cost or to minimize the cost for a fixed variance.

Both the problem can be formulated as below

(a) For fixed cost we have to:

$$\text{Minimize } V(\bar{y}_{st}) = \sum_{h=1}^{L} \frac{W_h^2 S_h^2}{n_h} - \sum_{h=1}^{L} \frac{W_h^2 S_h^2}{N_h}$$

$$\text{Subject to } C = c_0 + \sum c_h n_h$$

$$\text{and } N_h \geq n_h \geq 1$$

where c_0 denote the overhead cost and c_h denote the per unit measurement cost in the h^{th} stratum.

(b) For fixed variance we have to:

$$\text{Minimize } C_0 = \sum c_h n_h$$

$$\text{Subject to } \sum_{h=1}^{L} \frac{W_h^2 S_h^2}{n_h} - \sum_{h=1}^{L} \frac{W_h^2 S_h^2}{N_h} \leq v$$

$$\text{and } N_h \geq n_h \geq 1$$

where $C_0 = C - c_0 = \sum c_h n_h$ and v is the preassigned limit for $V(\bar{y}_{st})$. Using Cauchy's inequality the solution of both the problem can be obtained as

$$n_h = n \frac{W_h S_h / \sqrt{c_h}}{\sum W_h S_h / \sqrt{c_h}}$$ (See Cochran (1977))

The value of total sample size $n\,(=\sum n_h)$ for fixed cost and for fixed precision are given by

$$n=\frac{(C-c_0)\sum W_h S_h/\sqrt{c_h}}{\sum W_h S_h\sqrt{c_h}}$$

and $$n=\frac{(\sum W_h S_h\sqrt{c_h})(\sum W_h S_h/\sqrt{c_h})}{V+\frac{1}{N}\sum_{h=1}^{L} W_h S_h^2}$$

respectively.

where V is fixed according to the required precision.

1.7 CLUSTER SAMPLING:

In random sampling, we suppose that the population has been divided into a finite number of distinct and identifiable units called the sampling units. The smallest units into which the population can be divided are called the elements of the population, and a group of such elements is known as a cluster. The specified numbers of clusters obtain by a simple rule, the number of elements in a cluster should be small and the number of clusters should be large. A simple random of clusters is then obtained by the method of simple random sampling, with sampling unit as a cluster. This procedure of sampling is called cluster sampling.

There are two main reasons for using cluster sampling.

(1) Usually a complete list of the population units (sampling frame) is not available and therefore their use as sampling units is not possible.

(2) Even when a complete list (sampling frame) is available economic consideration compel us to take a larger sampling unit. For a given size of the sample, usually smaller sampling units give more precise results as compared to larger sampling units, but greater cost is involved in locating and approaching smaller units to measure them.

1.8 TWO STAGE SAMPLING:

In two stage sampling (also called sub-sampling) first the population units (called second stage units (ssu)) are grouped together to form first stage units (fsu). At the first stage an SRS of fsu is selected then at the second stage of sampling SRS of required number of ssu are selected to construct the estimator of the unknown population parameters.

This procedure can be generalized for three or more stages and is termed as multistage sampling.

1.9 THE GENERAL MATHEMATICAL PROGRAMMING PROBLEM (MPP):

Mathematical programming is concerned with finding optimal solutions to the problems of decision making under limited resources to meet the desired objectives.

The mathematical model of an MPP may be given as

$$\text{Maximize (or Minimize)} \quad z = f(x)$$

$$\text{Subject to} \quad g_i(\underline{x})[\leq,=,\geq] \; b_i; i = 1,2,...,m$$

$$\text{and} \quad \underline{x} \geq 0$$

where $\underline{x}' = (x_1, x_2, ..., x_n)$ is an n component vector of decision variables.

$f(\underline{x})$ and $g_i(\underline{x})$ are functions of $x_1, x_2, ..., x_n$, $b_i, i = 1,2,...,m$ are known constants.

Furthermore one and only one of the signs $[\leq,=,\geq]$ holds for each constraint.

1.10 MATHEMATICAL PROGRAMMING TECHNIQUES:

Depending upon the nature of objective function $f(\underline{x})$ and the constraints functions $g_i(\underline{x})$ and other restrictions on the decision variable, the MPP may be classified into two main classes.

(1) Linear Programming Problem(LPP)

(2) Non-Linear Programming Problem(NLPP)

If all the involved functions are linear the MPP is said be a linear programming problem (LPP), otherwise it will fall in the category of non-linear programming (NLPP). In other word in non-linear programming all the involved functions are not linear.

1.11 LINEAR PROGRAMMING PROBLEMS:

Linear programming problems involving two decision variables can easily be solved by the graphical method. The method also provides an insight into the concepts of simplex method, a powerful technique to solve the linear programming problems.

1.11.1 THE SIMPLEX COMPUTATIONAL PROCEDURE:

The stepwise algorithm of Simplex Method is given below:

Step 1: Convert the given LPP in the standard form "Minimize $Z = \underline{c}' \underline{x}$, s.t

$A\underline{x} = \underline{b}$ & $\underline{x} \geq 0$". Where A is of size $m \times n$.

Step 2: Select any m columns of A to form a square sub-matrix B of A such that

$|B| \neq 0$.

Step 3: Compute B^{-1} and then $B^{-1}\underline{b}$. If $B^{-1}\underline{b} \leq \underline{0}$.

Discard this B and go back to step 2. If no B is available with $|B| \neq 0$ &

$B^{-1}\underline{b} \geq \underline{0}$,STOP, the given LPP has no solution. Otherwise go to step 4 with $B^{-1}\underline{b} \geq \underline{0}$.

Step 4: Compute $\underline{c}_B' B^{-1}\underline{b}$, $B^{-1}A$ and $z_j - c_j = \underline{c}_B' B^{-1}\underline{a}_j - c_j$ where $\underline{c}_B'$ is the vector of the coefficient of basic variables in the objective function $z = \underline{c}'\underline{x}$. The basic variables are those whose corresponding columns are selected in B.

Step 5: Set up the starting simplex tableau as:

Tableau '0'

				$c_1,\ldots, c_j,\ldots, c_k,\ldots, c_n$
Row no.	$\underline{x}_B$	$\underline{c}_B$	*RHS*	$x_1,\ldots, x_j,\ldots, x_k,\ldots, x_n$
1 . . r . . m	x_{B_1} . . x_{B_r} . . x_{B_m}	c_{B_1} . . c_{B_r} . . c_{B_m}	$B^{-1}\underline{b} = \overline{\underline{b}}$	$B^{-1}A$
$m+1$	.....	....	$z = \underline{c}_B' B^{-1}\underline{b}$	$z_1 - c_1 \;\; z_j - c_j \;\; z_k - c_k \; z_n - z_n$

where $\underline{x}_B$ is the vector of basic variables.

Let $z_k - c_k = Maximum\{z_j - c_j\}$. If $z_k - c_k \leq 0$, STOP, The current solution is optimum. Otherwise go to step 6.

Step 6: Let $\underline{y}_k$ denote the k^{th} column of $B^{-1}A$, then $\underline{y}_k = B^{-1}\underline{a}_k$. If $\underline{y}_k \leq \underline{0}$, STOP, the given LPP has an unbounded solution. Otherwise go to step 7.

Step 7: Determine x_{B_r} as follows:

$$\frac{\bar{b}_r}{y_{rk}} = \underset{i}{Minimum}\left\{\frac{\bar{b}_i}{y_{ik}} / y_{ik} > 0\right\}$$

Where $\bar{b}_r$ is the $r-$th component of $B^{-1}\underline{b} = \underline{\bar{b}}$.

In the improved solution x_{B_r} will become non-basic and x_k will become basic in its place. Update the simplex tableau and go to step 5, repeat steps 5 to 7 until an 'STOP' indication is reached.

Updating:

i) Divide r^{th} row by y_{rk} to update it.

ii) For $i = 1,2,...,m$ and $i \neq r$ update the i^{th} row by adding '$-y_{ik}$' times the updated r^{th} row.

iii) Update the last row, the $(m+1)^{th}$ row, by adding '$-(z_k - c_k)$' times the updated r^{th} row.

Notes:

1) If no B with $|B| \neq 0$ and $B^{-1}\underline{b} \geq \underline{0}$, can be obtained (see steps 2 & 3) the given LPP has no solution.

2) When at step $2\ B = I_m$ the computations to set up Tableau '0' simplifies to a great extent because:

$B^{-1} = I_m$ and if $B^{-1}\underline{b} = \underline{b} \geq 0$, we have

$$\underline{c}'_B B^{-1}\underline{b} = \underline{c}'_B \underline{b} \quad , \quad B^{-1}A = A \quad , \quad z_j - c_j = \underline{c}'_B \underline{a}_j - c_j.$$

$$z = \underline{c}'_B B^{-1}\underline{b} = \underline{c}'_B \underline{b} \quad \text{and} \quad \underline{y}_k = B^{-1}\underline{a}_k = \underline{a}_k \text{ at step 6.}$$

Furthermore, if at the starting stage all the basic variables are slack or surplus variables we have $\underline{c}_B = \underline{0}$ and the computations are further simplified as

$$\underline{c}'_B B^{-1}\underline{b} = \underline{0} \quad , \quad z_j - c_j = \underline{c}'_B B^{-1}\underline{a}_j - c_j = -c_j$$

and $z = \underline{c}'_B B^{-1} b = 0$

When a readymade identity basis is not available we may use any of the following two methods to solve the LPP.

i) Two phase method (ii) Artificial basis technique (Big-M method)

1.12 TWO PHASE SIMPLEX METHOD:

Convert the given problem in standard form

$$\text{Minimize } z = \underline{c}' \underline{x}$$

$$\text{s.t } A\underline{x} = \underline{b}$$

$$\& \quad \underline{x} \geq 0$$

If any m columns of the $m \times n$ coefficient matrix A can be arranged to form an $m \times m$ identity matrix apply simplex method directly. Otherwise go to the phase-I of the Two Phase Method.

Phase I-

Step 1: Add artificial variables to complete the starting identity basis. Let $x_{n+1}, x_{n+2}, ..., x_{n+m}$ be the m artificial variables added. (We are assuming that no column of I_m is available in A).

Step 2: Solve the LPP

Minimize $S = x_{n+1} + x_{n+2} + ... + x_{n+m}$

Subject to $A\underline{x} + I\underline{x}_a = \underline{b}$

$$\underline{x}, \underline{x}_a \geq \underline{0}$$

Where $\underline{x}'a = (x_{n+1}, x_{n+2}, ..., x_{n+m})$ is the vector of artificial variables.

If all $z_j - c_j \leq 0$ but $\underline{x}_a \neq \underline{0}$, STOP, the given LPP has no solution.

Otherwise go to phase II.

Phase II:

Solve the LPP

$$\text{Minimize } z = \underline{c}' \underline{x}$$

$$\text{s.t } \underline{x}_B + B^{-1}N\underline{x}_N = B^{-1}\underline{b}$$

$$\underline{x} \geq \underline{0}$$

where $\underline{x} = \begin{pmatrix} \underline{x}_B \\ \underline{x}_N \end{pmatrix}$, $\underline{x}_B$ and $\underline{x}_N$ are the vectors of basic and non-basic variables in the final simplex tableau at Phase-I, step 2. We assume that all artificial variables are non-basic variables and are equal to zero and need not to be considered any more.

The Tableau '0' of Phase II can be obtained directly from the final tableau of Phase I as follows:

i) Delete the columns corresponding to the artificial variables.

ii) Insert original costs c_j at their proper places.

iii) Re-compute the last row elements that is, the value of the objective function and the $z_j - c_j$ elements using their usual formula.

After preparing the Tableau '0' of Phase-II if all $z_j - c_j \leq 0$, STOP, the present solution is the required optimal solution. Otherwise prepare the next tableau as usual.

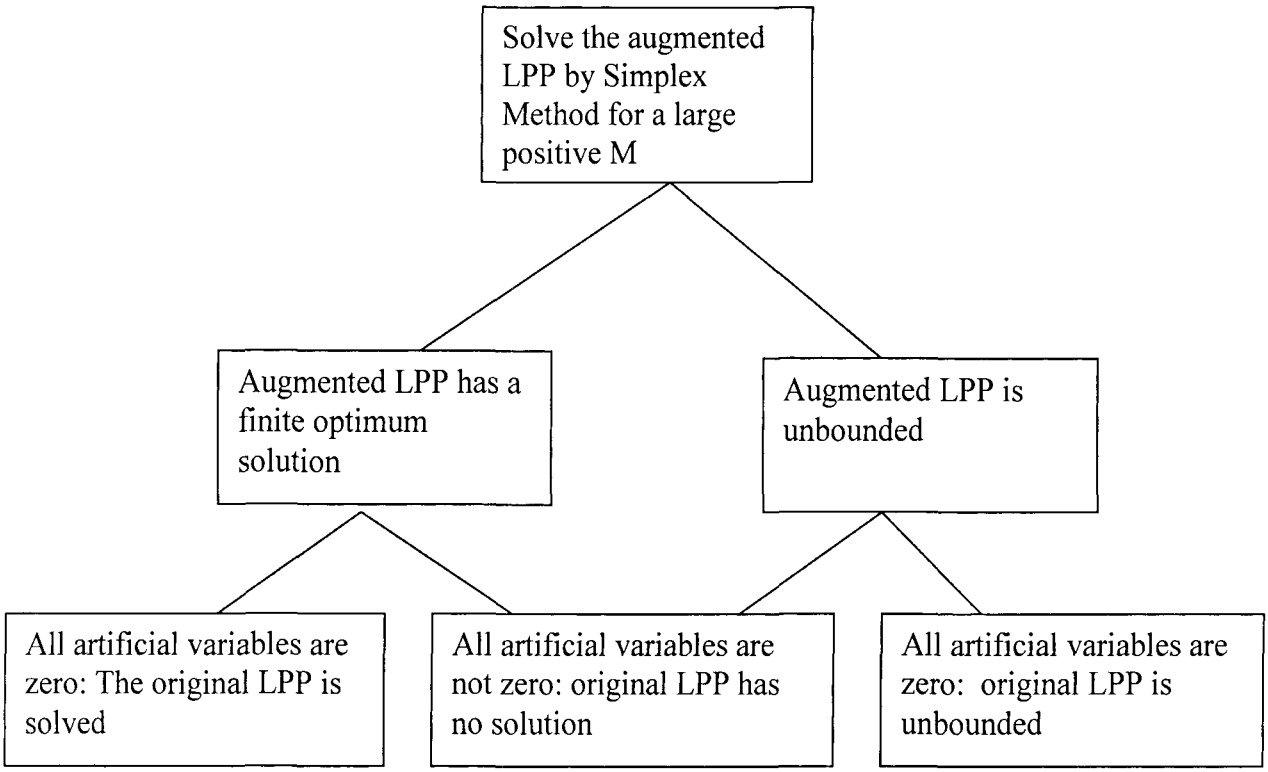
1.13 THE ARTIFICIAL BASIS TECHNIQUE (BIG-M METHOD):

In solving an LPP by Simplex Method if after converting the given LPP in standard form if the coefficient matrix A does not contain an $m \times m$ identity matrix that can be used as the starting basis matrix then we have to compute the inverse of the selected basis matrix B.

The computation of B^{-1} may be avoided by using an artificial identity basis matrix by introducing the required number of artificial variables in the constraint equations. This technique is known as the artificial basis technique or Big-M Method because in the objective function the artificial variables are added with a coefficient of '+M' for a minimization LPP and with a coefficient of '-M' for a maximization LPP, where M is a large positive number. The LPP with artificial variables is called the 'Augmented LPP'.

The 'Augmented LPP' has a coefficient matrix that contains an identity matrix that can be used as the starting basis to solve the 'Augmented LPP'.

When the simplex method is applied to the 'Augmented LPP' the following situations may arise. The conclusion in each situation is shown in the diagram below:



1.14 INTRODUCTION OF NLPP:

As discussed earlier any LPP can be solved by simplex method or its variations. The optimum solution lies at one of the extreme points of the convex feasible region .But in a non–linear programming problem (NLPP), The optimum solution can be found any where on the boundary of the feasible region or in its interior.

All LPP can be solved by using simplex method. Unlike to LPP there is no single technique that can solve every NLPP. Special techniques are

developed by exploiting the specific nature of the objective function, constraints and other

restriction on the decision variables.

NLPP may be classified in the following classes;

1- Quadratic programming

2-Geometric programming

3- Stochastic programming

4- Integer programming

5-Dynamic programming

6-Seperable programming etc. etc.

It is to be noted that all the above classes of MPP s are not disjoint and exhaustive .The details of all these programming techniques are beyond the scope of this dissertation. However some of the techniques that are used in the subsequent chapters are discussed in some details.

1.15 DYNAMIC PROGRAMMING TECHNIQUE:

The basic features which characterize the problems that can be handled by dynamic features may be described as follows.

1-The problem can be divided up into stages, with a policy decision required at each stage.

2- Each stage has a number of states associated with it.

3-The effect of the policy decision at each stage is to transform the current state into a state associated with the next stage.

4-Given the current state, an optimal policy for the remaining stages is independent of the policy adopted in previous stage.

5-The solution procedure is designed to find an optimal policy for the over all problem that is a prescription of the optimal policy decision at each stage for each of the possible states.

6-The solution procedure begins by finding the optimal policy for each state of the last stage.

7-A recursive relationship that identifies the optimal policy for stage n, given the optimal policy for stage (n +1) is available. This recursive relationship will always be of the form

$$f_n^*(s) = \max_{x_n}/\min_{x_n}\{f_n(s,x_n)\}$$

Therefore, finding the optimal policy decision when starting in state s at stage n requires finding the minimizing (or maximizing) value of x_n.The corresponding minimum cost (or maximum profit) is achieved by using this value of x_n and then following the optimal policy when starting in state x_n at stage $(n+1)$.

The recursive relationship is given its name because it keeps recurring as.

We move backward stage by stage. When the current stage number n is decreased by one, the new $f_n^*(s_n)$ function is derived by using the $f_{n+1}^*(s_{n+1})$ function that was just derived during the preceding iteration, and then this process keeps repeating. This property is emphasized in our next (and final) characteristic of dynamic programming. The precise form of the recursive relationship differs somewhat among differences dynamic programming problems. However, notation analogous to that introduced in the preceding section will continue to be used here, as summarized below.

N = number of stages

n = label for current stage $(n = 1,2,...,N)$

s_n = current state for stage n

x_n = decision variable for stage n

x_n^* = optimal value of x_n (given s_n)

$f_n(s_n, x_n)$ = contribution of stages $n, n+1,...,N$ to the objective function if the system starts in state s_n at stage n, the intermediate decision is x_n, and optimal decisions are made there after.

$$f_n^*(s_n) = f_n(s_n, x_n^*).$$

8. When we use this recursive relationship. The solution procedure moves Back-ward stage by stage-each time finding the optimal policy for stage-

until it finds the optimal policy at the initial stage. When this table is finally obtained for the initial stage$(n=1)$, the problem of interest is solved. Because the initial state is known, the initial decision is specified by x_1^* in this table. The optimal value of the other decision variables is then specified by the other tables in turn according to the state of the system that results from the preceding decisions.

1.16 NON - LINEAR PROGRAMMING PROBLEM (NLPP) AND KUHN- TUCKER (K-T) CONDITIONS:

An MPP in which all the involved functions are not linear is called a NLPP. In other words an MPP in which at least one of the involved functions is non-linear is called an NLPP. The following form of the NLPP is taken as the standard form for further discussions.

Maximize $f(\underline{x})$

Subject to $g_i(\underline{x}) \geq \underline{0}$, $(i=1,2,...,m)$_

and $\underline{x} \geq \underline{0}$

where $\underline{x}$ is the vector of decision variables $x_1, x_2, ..., x_n$. If the original problem is not in standard form it can be easily transformed into it by simple algebraic operations.

Feasible solutions:

An n-component vector $\underline{x}$ is called a feasible solution to NLPP if it satisfies $g_i(\underline{x}) \geq \underline{0}$, $i = 1,2,...,m$ and $\underline{x} \geq \underline{0}$. The set F of all feasible solutions to the NLPP is defined as:

$$F = \{\underline{x} \mid g_i(\underline{x}) \geq 0; i = 1,2,...,m; \underline{x} \geq \underline{0}\}.$$

Optimal solution:

An $\underline{x}^* \in F$ will be an optimal solution to the NLPP if $f(\underline{x}^*) \geq f(\underline{x})$ for all $\underline{x} \in F$. Kuhn and Tucker derived the following necessary conditions to be satisfied by the optimal solution $\underline{x}^*$ of an NLPP of size $m \times n$.

Let $\underline{x}^*$ be an optimal solution to the NLPP

Maximize $f(\underline{x})$

Subject to $g_i(\underline{x}) \geq \underline{0}$, $i = 1,2,...,m$

and $\underline{x} \geq \underline{0}$

Where the functions 'f' and 'g_i' ; $i = 1,2,...,m$ are differentiable. Assume that the constraint qualification holds. Then there exists a vector $\underline{u}^{*'}$ such that

$$\nabla_{\underline{x}} \phi(\underline{x}^*, \underline{u}^*) \leq \underline{0} \qquad \text{(i)}$$

$$\underline{x}^{*'} \nabla_{\underline{x}} \phi(\underline{x}^*, \underline{u}^*) = \underline{0} \quad \text{(ii)}$$

$$\nabla_{\underline{u}} \phi(\underline{x}^*, \underline{u}^*) \geq \underline{0} \quad \text{(iii)}$$

$$\underline{u}^{*'} \nabla_{\underline{u}} \phi(\underline{x}^*, \underline{u}^*) = \underline{0} \quad \text{(iv)}$$

$$\underline{x}^* \geq 0 \quad \text{(v)}$$

and

$$\underline{u}^* \geq \underline{0} \quad \text{(vi)}$$

Where

$$\phi(\underline{x}, \underline{u}) = f(\underline{x}) + \sum_{i=1}^{m} u_i g_i(\underline{x})$$

and

$\nabla \underline{x}$ and $\nabla \underline{u}$ represent the gradient vector of $\phi(\underline{x}, \underline{u})$ with respect to the components of $\underline{x}$ and $\underline{u}$ respectively.

Sufficiency of k-t conditions:

For NLPP's where $f(\underline{x})$ is pseudo concave and $g_i(\underline{x}); i = 1,2,...,m$ are quasi concave the above conditions are sufficient also. As a result, in such cases if we are able to find an $\underline{x}^*$ satisfying all the K-T conditions then $\underline{x}^*$ will be the required optimal solution of the given NLPP.

1.17 MATHEMATICAL PROGRAMMING TECHNIQUES IN SAMPLING:

Sampling, which is the selection of 'part' (sample) of an aggregate to represents the 'whole' (population), is used frequently in surveys almost in every walk of life. The purpose of sample survey is to obtain information about the population which is defined according to the aims and objectives of the survey. The information on population is based on sample data, size of sample, the sampling scheme, number of strata and stratum boundaries etc. these decisions are very important. For example, the decision regarding the size of sample to be selected is important because too large sample implies waste of resources and too small sample diminishes the utility of the results obtained. Therefore, the problem of deriving the statistical information on population characteristics can be formulated as an optimization problem of minimizing the cost of survey subject to the restriction that the loss of precision must be within a certain prescribed limit or alternately minimizing the loss in precision subject to the restriction that the cost of the survey remain within the given budget.

Noted statistician C. R. Rao in the preface to Arthanari and Hodges (1981) advocated the use of Mathematical Programming Techniques in the problem of optimization arising in statistics in the following words.

"All statistical procedures are, in the ultimate analysis, solutions to suitably formulated optimization problems. Whether it is designing a scientific experiment, or planning a large scale survey for collection of data, or choosing a stochastic model to characterize observed data, or drawing inference from available data, such as estimation, testing of hypothesis, and decision making, one has to choose an objective function and minimize or maximize it subject to given constraints on unknown parameters and inputs such as the costs involved. The classical optimization methods based on the differential calculus are too restrictive, and are either inapplicable or difficult to apply in many situations that arise in statistical work. This together with the lack of suitable numerical algorithms for solving optimizing equations has placed severe limitation on the choice of objective functions and constraints and led to the development and use of some inefficient statistical procedures.

Attempts have therefore been made during the last three decades to find other optimization techniques that have wider applicability and can be easily implemented with the available computing power. One such technique that has the potential for increasing the scope for application of efficient statistical methodology is "Mathematical Programming".

CHAPTER-II

OPTIMUM STRATIFICATION: THE CLASSICAL APPROACH

2.1 INTRODUCTION:

After deciding the number of strata the sampler has to fix the strata boundaries. If the frequency distribution of the main variable y under study is known then the best criterion for stratification is the frequency distribution of y itself. If the frequency distribution of y is not know some other auxiliary variable x, highly correlated with y, and whose frequency distribution is known may be used to determine optimum strata boundaries(OSB). This variable x is known as the stratification variable.

2.2 AN OVERVIEW OF THE PROBLEM OF DETERMINING THE OSB:

The basic consideration involved in the determination of OSB is that the strata should be internally as homogenous as possible, that is, the stratum variances σ_h^2 should be as small as possible. When a single characteristic y is under study and its frequency distribution is

available, the OSB can be determined by cutting the range of this distribution at suitable points. The problem of determining the OSB was first discussed by Dalenius (1950) when the study variable itself is used as stratification variable. He presented a set of minimal equations whose solution could provide the OSB. Unfortunately these equations could not usually be solved because of their implicit nature. Attempts have been made by several authors to obtain the OSB using various methods. Given the number of strata, Dalenius and Gurney (1951) suggested that the strata boundaries should be determined such that the products $W_h\sigma_h$ remain constant. Mahalanobis (1952) and Hansen, et al. (1953) have suggested that the strata boundaries should be determined such that $W_h\mu_h$ remain constant, where μ_h is the stratum mean of the h^{th} stratum. Aoyoma (1954) suggested an approximate rule and recommended to make strata of equal width. Ekman (1959) determined the strata boundaries with constant $W_h(y_h - y_{h-1})$. Dalenius and Hodges (1959) recommended to construct the equally spaced strata boundaries on the cumulative $\sqrt{f(y)}$ scale, where $f(y)$ denote the frequency function of y. Sethi (1963) proposed a method to work out the boundaries given by the

equations $\frac{(y_h - \mu_h)^2 + \sigma_h^2}{\sigma_h} = \frac{(y_{h+1} - \mu_{h+1})^2 + \sigma_{h+1}^2}{\sigma_{h+1}}$ for a standard continuous distribution resembling the study population.

In a comparison on some of the classical approximate methods for working out OSB, the methods of Ekman, and Dalenius and Hodge work consistently well (see Cochran (1961), Hess,et al.(1966) and Murthy (1967)). But the later is more convenient and easier to apply (see Nicoloni (2001)). Unnithan (1978) suggested an iterative method using Shanno's Modified Newton method for determining the OSB that leads to a local minimum of the variance for Neyman allocation provided a suitable initial solution is chosen. The procedure is proved to be faster than the Dalenius and Hodges iterative procedure. Later on Unnithan and Nair (1995) gave a method of selecting an appropriate starting point for modified Newton method that may lead to a global minimum of the variance.

Lavallee and Hidiroglou (1988) proposed an algorithm to construct stratum boundaries for a power allocated stratified sample of non-certainty sample units. Hidiroglou and Srinath (1993) presented a more general form of the algorithm, which by assigning different values to operating parameters yields a power allocation, a Neyman

allocation, or a combination of these allocation. Sweet and Sigman (1995 a, b) and Rivest (2002) reviewed these methods and confined their discussion to the use of the Lavallee and Hidiroglou algorithm with Neymann allocation. Detlefsen and Veum (1991) investigated the Lavallee and Hidiroglou algorithm for several strata and observed that the algorithm's convergence was slow or non-existent. They also found that different starting points leads to different OSB for the same population.

Niemiro (1999) proposed a random search method for the stratification problem but his algorithm did not guarantee a global optimum. Furthermore, it may go wrong for large populations, as it requires too many iterations (seeKozak (2004)).

Nicolini (2001) suggested a method, named Natural Class Method (NCM), as an alternative to popular Dalenius and Hodges method but neither method was proved to be more efficient than other.

Rivest (2002) and Lednicki and Wieczorkowski (2003) presented a method of stratification using the simplex method of Nelder and Mead (1965). Later Kozak (2004) presented the modified random search algorithm for finding OSB. Although the Kozak algorithm was faster

and efficient as compared to Rivest, and Lednicki and Wieczorkowski but does not guarantee a global optimum.

Biihler and Deutler (1975) formulated the problem of determining OSB as an optimization problem and developed a computational technique to solve the problem using dynamic programming. This approach is also used by Lavallee (1987,1988) for determining the OSB which would divide the population domain of two stratification variables into distinct subsets such that the precision of the estimates are maximized.

Khan, et al. (2002) considered the problem of finding OSB as an equivalent problem of determining Optimum Strata Width (OSW). The authors formulated the problem as a Mathematical Programming Problem (MPP). They solved the MPP using Dynamic Programming Technique (DPT) that gives exact solution. They applied DPT to work out OSB for the populations having uniform and right triangular distributions. Later on Khan, et al. (2005) extended their technique for determining the OSB for an exponential study variable and Nand, N. et al (2008) used the same technique for determining the OSB for log-normal distribution.

2.3 THE CLASSICAL APPROACH:

Assuming that the frequency distribution of the main variable y is known that is, y itself can be used as stratification variable, Dalenius (1957) worked out the best stratum boundaries under proportional and Neyman allocations.

In this section the Dalenius (1957) approach under Neyman allocation

and the 'Approximate Strata Boundaries' worked out by Dalenius and Hodge (1959) are discussed.

Let L strata are to be constructed and y_0 and y_L be the smallest and largest values of y, respectively, in the population. Then the problem is to find $L-1$ stratum boundaries $y_1, y_2, ..., y_{L-1}$ such that

$$V(\bar{y}_{st}) = \frac{1}{n}\left(\sum_{h=1}^{L} W_h S_h\right)^2 - \frac{1}{N}\sum_{h=1}^{L} W_h S_h^2 \qquad (2.3.1)$$

is minimum. If f.p.c is ignored, it is sufficient to minimize,

$\frac{1}{n}\left[\sum_{h=1}^{L} W_h S_h\right]^2$ or subsequently to minimize $\sum_{h=1}^{L} W_h S_h$.

Since y_h appears in this sum only in the terms $W_h S_h$ and $W_{h+1} S_{h+1}$, we have

$$\frac{\partial}{\partial y_h}\left(\sum W_h S_h\right)=\frac{\partial}{\partial y_h}\left(W_h S_h\right)+\frac{\partial}{\partial y_h}\left(W_{h+1} S_{h+1}\right)$$

If $f(y)$ is the frequency function of y, then we may express

$$W_h = \int_{y_{h-1}}^{y_h} f(t)dt, \qquad \frac{\partial W_h}{\partial y_h} = f(y_h) \tag{2.3.2}$$

and

$$W_h {S_h}^2 = \int_{y_{h-1}}^{y_h} t^2 f(t)dt - \frac{\left(\int_{y_{h-1}}^{y_h} tf(t)dt\right)^2}{\int_{y_{h-1}}^{y_h} f(t)dt} \tag{2.3.3}$$

Differentiating (2.3.3) w.r.t. y_h, we get

$${S_h}^2 \frac{\partial W_h}{\partial y_h} + 2W_h S_h \frac{\partial S_h}{\partial y_h} = {y_h}^2 f(y_h) - 2y_h \mu_h f(y_h) + \mu_h^2 f(y_h) \tag{2.3.4}$$

Where μ_h is the stratum mean of y in stratum h. Using (2.3.2) we may add ${S_h}^2 \partial W_h / \partial y_h$ to the LHS, and the equal quantity $S_h^2 f(y_h)$ to the RHS of (2.3.4) and dividing by $2S_h$,we get:

$$\frac{\partial(W_h S_h)}{\partial y_h} = S_h \frac{\partial W_h}{\partial y_h} + W_h \frac{\partial S_h}{\partial y_h} = \frac{1}{2} f(y_h) \frac{(y_h - \mu_h)^2 + {S_h}^2}{S_h} \tag{2.3.5}$$

Similarly we have:

$$\frac{\partial(W_{h+1}S_{h+1})}{\partial y_h} = \frac{1}{2} f(y_h) \frac{(y_h - \mu_{h+1})^2 + S_{h+1}^2}{S_{h+1}} \qquad (2.3.6)$$

The calculus equations for y_h can now be given as:

$$\frac{(y_h - \mu_h)^2 + S_h^2}{S_h} = \frac{(y_h - \mu_{h+1})^2 + S_{h+1}^2}{S_{h+1}} \qquad (h = 1,2,...L-1) \qquad (2.3.7)$$

(See Dalenius(1950)).

Unfortunately, these equations are implicit in y_h and no technique is available to work out their exact solution. It is because both μ_h and S_h depend on y_h.

If the class intervals in the original distribution of y are of unequal length, a slight change is needed. When the interval changes from one of length d to one of length ud, the value of $\sqrt{f}$ for the second interval is multiplied by $\sqrt{u}$ when forming $cum\sqrt{f}$.

Another method, proposed by Sethi (1963), is to work out the boundaries given by the calculus equations (2.3.7) for a standard continuous distribution resembling the study population. For the normal and various χ^2 distributions, Sethi has tabulated the optimum

boundaries for Neyman, equal, and proportional allocations for $L \leq 6$. If one of these distributions seems to approximate the distribution of stratification variable then OSB can be read from Sethi's tables.

Two further approximate methods require some trial and error. From relations

$$\sqrt{12}\sum_{h=1}^{L} W_h S_h \approx \sum_{h=1}^{L} f_h (y_h - y_{h-1})^2 \approx \sum_{h=1}^{L} (z_h - z_{h-1})^2 = \sum_{h=1}^{L} A_h^2 \tag{2.3.8}$$

the Dalenius-Hodges rule is roughly equivalent to making $W_h S_h$ constant, as conjectured earlier by Dalenius and Gurney (1951). A similar rule is that of Ekman (1959), who makes $W_h(y_h - y_{h-1})$ constant.

In comparisons on some theoretical and eight study populations, Cochran (1961) found that the $cum\sqrt{f}$ rule and the Ekman rule worked consistently well (the Sethi method was not tried). In study of United States hospital bed capacity, whose distribution resembles χ^2 with 1 degree of freedom, Hess, Sethi and Balakrishnan (1966) found the Ekman method is slightly superior to $cum\sqrt{f}$ and Sethi's for

$L > 2$, while Murthy (1967) also reports good performance by Ekman's rule .

The relations (2.3.8) have an interesting consequence. If $W_h S_h$ is constant, Neymann allocation gives a constant sample size $n_h = \frac{n}{L}$ in all strata. For the approximate methods, the comparisons that have been made suggest that the simple rule $n_h = \frac{n}{L}$ is satisfactory.

Usually the frequency distribution of y is not known. In practice, some other variables x is used as stratification variable such as the value of y at a recent census. Dalenius (1975) developed equations for the boundaries of x that minimize $\sum W_h S_{yh}$, given knowledge of the regression of y on x. If this regression is non-linear, these boundaries may differ considerably from those that are optimum when x itself is the variable to be measured. These equations indicate that if the regression of y on x is linear and the correlation between y and x is high in all strata the two sets of boundaries should be nearly the same.

Let

$$y = \alpha + \beta x + e \qquad (2.3.9)$$

where E(e)=0 for all x and e, x are uncorrelated. If the variance of e within h^{th} stratum is denoted by S_{eh}^2 then the x-boundaries that make $V(\bar{y}_{st})$ minimum satisfy the equations (See Dalenius, (1975)).

$$\frac{\beta^2\left[(x_h-\mu_{xh})^2+S^2{}_{xh}\right]+2S^2{}_{eh}}{\beta S_{xh}\sqrt{1+S_{eh}^2/\beta^2 S_{xh}^2}}$$

$$=\frac{\beta^2\left[(x_h-\mu_{x,h+1})^2+S_{x,h+1}^2\right]+2S_{e,h+1}^2}{\beta S_{x,h+1}\sqrt{1+S_{e,h+1}^2/\beta^2 S_{x,h+1}^2}} \qquad (.3.10)$$

If $S_{eh}^2/\beta^2 S_{xh}^2$ is small for all h, these equations reduce to the form (2.3.7) that gives optimum boundaries for x.

2.4 APPROXIMATE OPTIMUM STRATA BOUNDARIES:

Various authors worked out the approximate optimum strata boundaries. Dalenius and Hodge (1959) minimized $\sum W_h S_h$ and obtained approximate strata boundaries assuming that $f(y)$ is approximately rectangular within each stratum.

Let $Z(y)=\int_{y_0}^{y}\sqrt{f(t)}\,dt$

Then $W_h \cong \int_{y_{h-1}}^{y_h} f(t)dt = f_h(y_h - y_{h-1})$ (2.4.1)

$$S_h \cong \frac{1}{\sqrt{12}}(y_h - y_{h-1}) \quad (2.4.2)$$

and $Z_h - Z_{h-1} \cong \int_{y_{h-1}}^{y_h} \sqrt{f(t)}dt = \sqrt{f_h}(y_h - y_{h-1})$ (2.4.3)

where f_h is the 'constant' value of $f(y)$ in stratum h. (2.4.1), (2.4.2), and (2.4.3) give

$$\sqrt{12}\sum_{h=1}^{L} W_h S_h \cong \sum_{h=1}^{L} f_h(y_h - y_{h-1})^2 \cong \sum_{h=1}^{L}(Z_h - Z_{h-1})^2 \quad (2.4.4)$$

Since $(Z_L - Z_0)$ is fixed, the LHS of (2.4.4) will be minimum by making $(Z_h - Z_{h-1})$ constant.

Given $f(y)$, the rule is to form the cumulative $\sqrt{f(y)}$ and choose the y_h so that they create equal intervals on the cum $\sqrt{f(y)}$ scale.

2.5 THE MINIMUM VARIANCE STRATIFICATION (MVS):

Dalenius (1959) worked out the approximate strata boundaries to minimize the variance of the stratified sample mean. This approach is discussed below.

First approximation:

Consider the transformation

$$y(u) = \int_{-\infty}^{u} \sqrt{f(t)}dt \qquad (2.5.1)$$

When $u \to \infty$, $y(u)$ approaches an upper bound H. The roots $x'_1 \ldots x'_h \ldots x'_{L-1}$ of the following equations

$$y(u) = \frac{h}{L}H, \qquad h = 1 \ldots L-1 \qquad (2.5.2)$$

are taken as the (first)approximations, for large L,to the points $x_1 \ldots x_h \ldots x_{L-1}$ satisfying equation (2.3.7).

Justification:

This approximation may be derived by the following heuristic argument. When L is large, the strata will be narrow, and each will have an approximately rectangular distribution, so that $\sqrt{12}\sigma_h = x_h - x_{h-1}$. Then, by the mean value theorem there exists a value f_h of f in the h^{th} stratum, such that

$$\sqrt{12}\sum W_h\sigma_h = \sum[\sqrt{f_h}(x_h - x_{h-1})]^2 \cong \sum[y_h - y_{h-1}]^2 \qquad (2.5.3)$$

The last sum is minimized by making $y_h - y_{h-1}$ = constant. A rigorous proof has been given by Dalenius and Hodges (1959).

Adoption to numerical calculations:

Let density $f(x)$ be stratified into L strata. Two consecutive strata are specified by x_{h-1}, x_h and x_{h+1}. In order to simplify the formulas, we will denote these points of stratification by $x_{h-1} = x_g, x_h = x_h$ and $x_{h+1} = x_i$.

The interval x_g, x_h corresponds to the h^{th} stratum, and x_h, x_i to the i^{th} stratum.

Define

$$I_p(u) = \int_{-\infty}^{u} t^p f(t)dt \tag{2.5.4}$$

The conditional means μ_h, μ_i and variances σ_h^2, σ_i^2 of the two strata can be expressed in terms of $I_p(u)$ as

$$\mu_h = \frac{\int_{x_g}^{x_h} tf(t)dt}{\int_{x_g}^{x_h} f(t)dt} = \frac{I_1(x_h) - I_1(x_g)}{I_0(x_h) - I_0(x_g)} \tag{2.5.5}$$

$$\sigma_h^2 = \frac{\int_{x_g}^{x_h} t^2 f(t)dt}{\int_{x_g}^{x_h} f(t)dt} - \mu_h{}^2 = \frac{I_2(x_h) - I_2(x_g)}{I_0(x_h) - I_0(x_g)} - \mu_h{}^2 \qquad (2.5.6)$$

Now define

$$J_{ph} = I_p(x_h) - I_p(x_g) \qquad (2.5.7)$$

Thus

$$\mu_h = \frac{J_{1h}}{J_{0h}} \qquad (2.5.8)$$

$$\sigma_h^2 = \frac{J_{2h}}{J_{0h}} - \frac{J_{1h}^2}{J_{2h}^2} = \frac{J_{0h}J_{2h} - J_{1h}^2}{J_{0h}^2} \qquad (2.5.9)$$

The condition given by equation (2.3.7) may now be expressed in terms of J_{ph} and J_{pi} as follows

$$\frac{J_{2h} - 2x_h J_{1h} + x_h{}^2 J_{0h}}{\sqrt{J_{0h}J_{2h} - J_{1h}{}^2}} - \frac{J_{2i} - 2x_h J_{1i} + x_h{}^2 J_{0i}}{\sqrt{J_{0i}J_{2i} - J_{1i}{}^2}} = 0 \qquad (2.5.10)$$

For simplicity, this expression may be written as

$$A_h - B_h = \Delta_h = 0 \qquad (2.5.11)$$

The set $[x_h]$ satisfying the thumb rule $W_h\mu_h$ = constant corresponds to MVS. If we substitute any other values, say x_h', we will denote the

left and right side of equation (2.5.11) by A_h' and B_h' respectively and the difference by Δ_h'.

Second approximation:

In general, we may not expect the set $[x_h']$ derived from equation (2.5.2) to satisfy equation (2.5.11). Thus there is need for some method for adjusting the initial set $[x_h']$ into a set $[x_h'']$ which then can be checked in equation (2.5.11) etc.

2.6 A NUMERICAL EXAMPLE:

Consider a rectangular distribution $f(x)=1$. With no loss of generality, we may assume $0 \le x \le b$. For this distribution we have:

$$I_0 = J_0 = b$$

$$I_1 = J_1 = \frac{1}{2}b^2 \qquad (2.6.1)$$

$$I_2 = J_2 = \frac{1}{3}b^3$$

Substituting these values in

$$A = \frac{J_2 - 2bJ_1 + b^2 J_0}{\sqrt{J_0 J_2 - J_1^{\,2}}} \tag{2.6.2}$$

which is A_h for $h = 1$,we get

$$A = \frac{2}{\sqrt{3}} b \sim 1.15b \tag{2.6.3}$$

i.e. 'A' changes at about 1.15 times the rate at which the interval length $(0,b)$ changes.

As the next step, consider that this rectangular distribution is divided into $L = 2$ strata , at a point x_1' ,$0 < x_1' < b$, chosen arbitrarily. This point x_1' specifies A_1' and B_1'. Applying the above result , we realize that changing x_1' by one unit will change A_1' and B_1' by approximately 1.15 units each and the difference Δ_1' by approximately 2.3 units .It seems reasonable to determine a second point x_1'' of stratification by setting Δ_1'' equal to zero, where

$$\Delta_1'' = \Delta_1' + (x_1'' - x_1')\frac{4}{\sqrt{3}} \tag{2.6.4}$$

The solution of $\Delta_1'' = 0$ is given by

$$x_1'' = x_1' - \frac{\sqrt{3}}{4}\Delta_1' \tag{2.6.5}$$

The point x_1'' may reasonably be expected to be superior to x_1' as an approximation to the MVS point x_1.

Now consider the general case with L strata. For three consecutive strata, with indices g, h and i, we have

$$\left.\begin{aligned} \frac{\partial A_h'}{\partial x_g'} = \frac{\partial B_h'}{\partial x_h'} = -\frac{2}{\sqrt{3}} \\ \frac{\partial A_h'}{\partial x_h'} = \frac{\partial B_h'}{\partial x_i'} = \frac{2}{\sqrt{3}} \end{aligned}\right\} \tag{2.6.6}$$

while all expressions of the type $\partial A_h' / \partial x_i'$, $\partial B_h' / \partial x_g'$ etc. are equal to zero. From $\Delta_h' = A_h' - B_h'$ we derive analogous expressions for $\partial \Delta_h / \partial x_h'$ etc.

These values will now be used in the following way. We have a set $[x_h']$ with corresponding Δ_h'-values. We want to find a set $[x_h'']$ with $\left|\Delta_h''\right| < \left|\Delta_h'\right|$.

By the mean value theorem we have

$$\Delta_h^{''} = \Delta_h^{'} + (x_g^{''} - x_g^{'})\frac{\partial \Delta_h^{'}}{\partial x_g^{'}} + (x_h^{''} - x_h^{'})\frac{\partial \Delta_h^{'}}{\partial x_h^{'}}$$

$$+ (x_i^{''} - x_i^{'})\frac{\partial \Delta_h^{'}}{\partial \Delta_i^{'}}, \qquad h = 1,...,g,h,i...L-1, \tag{2.6.7}$$

where we put $x_0'' = x_0'$ and $x_L'' = x_L'$. Solving this system for x_h'' gives the set wanted.

The $(L-1)\times(L-1)$ matrix M of $\partial\Delta_h^{'}/\partial_j^{'}$ is under the approximated rectangular distribution is

$$\mathrm{M} = \frac{2}{\sqrt{3}}\begin{Vmatrix} 2 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & & \ldots & -1 & 2 \end{Vmatrix} \tag{2.6.8}$$

In the first and last rows we assumed the rectangular distribution with finite range which is limited at one end by the point of very large absolute value. The more the number of L, the better the approximation by the rectangular distribution. The inverse of M is

$$M^{-1} = \frac{\sqrt{3}}{2L}\begin{Vmatrix} L-1 & L-2 & \cdots & 2 & 1 \\ L-2 & 2(L-2) & \cdots & 4 & 2 \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & 6 & 3 \\ 3 & 6 & \cdots\cdots & \cdots\cdots & \\ 2 & 4 & \cdots & 2(L-2) & L-2 \\ 1 & 2 & \cdots & (L-2) & L-1 \end{Vmatrix} \qquad (2.6.9)$$

Thus, the new set $[x_h'']$ is found by computing

$$\left.\begin{aligned} x_1'' &= x_1' - \frac{\sqrt{3}}{2L}[(L-1)\Delta_1' + (L-2)\Delta_2' + \ldots + \Delta_{L-1}'] \\ x_2'' &= x_2' - \frac{\sqrt{3}}{2L}[(L-1)\Delta_1' + 2(L-2)\Delta_2' + \ldots + 2\Delta_{L-1}'] \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_{L-1}'' &= x_{L-1}' - \frac{\sqrt{3}}{2L}[\Delta_1' + 2\Delta_2' + \ldots + (L-1)\Delta_{L-1}'] \end{aligned}\right\} \qquad (2.6.10)$$

If necessary, the procedure may be repeated to give a third set $[x_h''']$ etc.

As the final step in the adjustment procedure, it is assumed that the procedure discussed above for the rectangular case, holds reasonably well also for other cases.

CHAPTER III

THE PROBLEM OF STRATIFICATION AS A MATHEMATICAL PROGRAMMING PROBLEM

3.1 INTRODUCTION:

The method of choosing the best boundaries that make strata internally homogeneous as far as possible is known as optimum stratification. To achieve this, the strata should be constructed in such a way that the strata variances for the characteristic under study be as small as possible. If the frequency distribution of the study variable x is known the Optimum Strata Boundaries (OSB) could be obtained by cutting the range of the distribution at suitable points. If the frequency distribution of x is unknown, it may be approximated from the past experience or some prior knowledge obtained at a recent study. In this chapter the general problem of finding the OSB is formulated as a Mathematical programming problem (MPP) that seeks minimization of the variance of the estimated population parameter under Neyman allocation subject to the constraint that the sum of the widths of all the strata is equal to the range of the distribution. The formulated MPP

turns out to be a multistage decision problem that can be approached by dynamic programming technique.

3.2 THE FORMULATION:

Let X be a random study variable with distribution function $F(x), a \le x \le b$. To estimate the population mean μ by a stratified sample the range of X is partitioned into L strata defined by $[a, x_1], (x_1, x_2], ..., (x_{L-1}, b]$ such that

$$a = x_0 \le x_1 \le x_2 \le, ..., x_{L-1} \le x_L = b. \tag{3.2.1}$$

Suppose that from stratum $h(h = 1,2,..., L)$, which contains N_h units, a sample of size n_h is obtained. Let y_{hj} denote the value of the $j^{th}(j = 1,2,..., n_h)$ units in the h^{th} stratum. Then the stratified sample mean $\bar{x}_{st} = \sum_{h=1}^{L} W_h \bar{x}_h$ will be an unbiased estimate of the population mean μ with a variance

$$V(\bar{x}_{st}) = \sum_{h=1}^{L} W_h \sigma_h^2 \left(\frac{W_h}{n_h} - \frac{1}{N} \right), \tag{3.2.2}$$

where $W_h = N_h / N$ and $\bar{x}_h = \frac{1}{n_h} \sum_{j=1}^{n_h} y_{hj}$.

For the h^{th} stratum we have

$$W_h(x_{h-1},x_h)=\int_{x_{h-1}}^{x_h} dF(x)dx,$$

$$\mu_h(x_{h-1},x_h)=\frac{1}{W_h}\int_{x_{h-1}}^{x_h} xdF(x)dx$$

and $$\sigma_h^2(x_{h-1},x_h)=\frac{1}{W_h}\int_{x_{h-1}}^{x_h}(x-\mu_h)^2 dF(x)dx.$$

Using these values of W_h,μ_h and σ_h^2, the RHS of (3.2.2) can be expressed as a function of x_h and n_h, that is,

$$V(\bar{x})_{st}=V(\bar{x}_{st}\mid x_1,...,x_{L-1},n_1,...,n_L).$$

If n_h are fixed, the objective of the optimum stratification is to determine stratum boundary points $(x_1,...,x_{L-1})$ such that $V(\bar{x}_{st})$ is minimum. Further, if the sampling ratio n_h/N_h are small or the sampling is with replacement and the population mean is estimated under Neyman allocation $(n_h=n\cdot W_h\sigma_h/\sum_{h=1}^{L}W_h\sigma_h)$, then the problem of determining OSB reduces to

$$\text{Minimize}\left\{\sum_{h=1}^{L}W_h\sigma_h \mid a=x_0\le x_1\le x_2\le,...,\le x_{L-1}\le x_L=b\right\}. \quad (3.2.3)$$

Let $f(x)$ denote the frequency function and x_0 and x_L be the smallest and largest values of x. Then (3.2.3) is equivalent to the problem of determining the strata boundaries to cut up the range

$$x_L - x_0 = d \text{ (say)} \tag{3.2.4}$$

at $(L-1)$ intermediate points $x_1 \leq x_2 \leq,...,\leq x_{L-1}$ such that $\sum_{h=1}^{L} W_h \sigma_h$ is minimum.

Where

$$W_h = \int_{x_{h-1}}^{x_h} f(x)dx, \tag{3.2.5}$$

$$\sigma_h^2 = \frac{1}{W_h} \int_{x_{h-1}}^{x_h} x^2 f(x)dx - \mu_h^2, \tag{3.2.6}$$

with $$\mu_h = \frac{1}{W_h} \int_{x_{h-1}}^{x_h} xf(x)dx. \tag{3.2.7}$$

Using (3.2.5),(3.2.6) and (3.2.7), W_h, σ_h^2 and μ_h could be expressed as a function of x_h and x_{h-1}. Hence the objective function in (3.2.3) could also be expressed as a function of x_h and x_{h-1} only and the problem (3.2.3) reduces to

$$\text{Minimize} \quad \sum_{h=1}^{L} f_h(x_{h-1}, x_h),$$

$$\text{Subject to} \quad a = x_0 \leq x_1 \leq x_2 \leq,...,\leq x_{L-1} \leq x_L = b. \tag{3.2.8}$$

Let $y_h = x_h - x_{h-1} \geq 0$ denote the width of the $h^{th}(h = 1,2,...,L)$ stratum.

With the above definition of y_h, the range of the distribution given in (3.2.4) may be expressed as the function of the stratum widths as:

$$\sum_{h=1}^{L} y_h = \sum_{h=1}^{L} (x_h - x_{h-1}) = x_L - x_0 = d. \qquad (3.2.9)$$

The k^{th} stratification point $x_k; (k = 1,2,...,L-1)$ is then expressed as:

$$x_k = x_0 + y_1 + y_2 + ... + y_k$$

$$= x_{k-1} + y_k,$$

which is a function of k^{th} stratum width and $(k-1)^{th}$ stratum boundary.

Adding (3.2.9) as a new constraint, the problem (3.2.8) can be treated as an equivalent problem of determining OSW as:

$$\text{Minimize} \quad \sum_{h=1}^{L} f_h(y_h, x_{h-1}),$$

$$\text{Subject to} \quad \sum_{h=1}^{L} y_h = d,$$

$$\text{and} \qquad y_h \geq 0; h = 1,2,...L. \qquad (3.2.10)$$

Initially, x_0 is known. Therefore, the first term, that is, $f_1(y_1, x_0)$ in the objective function of MPP (3.2.10) is a function of y_1 alone. Once y_1 is known, the next stratification point $x_1 = x_0 + y_1$ will be known and the second term in the objective function $f_2(y_2, x_1)$ will become

a function of y_2 alone. Thus, stating the objective function as a function of y_h alone, we may rewrite the MPP (3.2.10) as:

$$\text{Minimize } \sum_{h=1}^{L} f_h(y_h),$$

$$\text{subject to } \sum_{h=1}^{L} y_h = d,$$

$$\text{and } \quad y_h \geq 0; h = 1,2,...L. \qquad (3.2.11)$$

3.3 THE SOLUTION USING DYNAMIC PROGRAMMING TECHNIQUE:

The problem (3.2.11) is a multistage decision problem in which the objective function and the constraints are separable functions of y_h which allow us the use of dynamic programming technique.

Consider the following subproblem of (3.2.11) for $k(< L)$ strata.

$$\text{Minimize } \sum_{h=1}^{k} f_h(y_h)$$

$$\text{subject to } \sum_{h=1}^{k} y_h = d_k$$

$$\text{and } y_h \geq 0;\ h = 1,2,\ldots,k. \qquad (3.3.1)$$

where $d_k = d$ is the total width available for division into k strata.

Note that $d_k = d$ for $k = L$

Also

$$
\begin{aligned}
d_k &= y_1 + y_2 + \ldots + y_k \\
d_{k-1} &= y_1 + y_2 + \ldots + y_{k-1} = d_k - y_k \\
d_{k-2} &= y_1 + y_2 + \ldots + y_{k-2} = d_{k-1} - y_{k-1} \\
&\vdots \qquad\qquad \vdots \\
d_2 &= y_1 + y_2 = d_3 - y_3 \\
d_1 &= y_1 = d_2 - y_2.
\end{aligned}
$$

Let $f(k,d_k)$ denotes the minimum value of the objective function (3.3.1), that is,

$$
f(k,d_k) = \min\left[\sum_{h=1}^{k} f_h(y_h) \mid \sum_{h=1}^{k} y_h = d_k, \text{ and } y_h \geq 0; h = 1,2,\ldots,k\right].
$$

With the above definition of $f(k,d_k)$ the problem (3.2.11) is equivalent to finding $f(L,d)$ recursively by finding $f(k,d_k)$ for $k = 1,2,\ldots,L$ and $0 \leq d_k \leq d$.

We can write

$$
f(k,d_k) = \min\left[f_k(y_k) + \sum_{h=1}^{k-1} f_h(y_h) \mid \sum_{h=1}^{k-1} y_h = d_k - y_k, \text{and } y_h \geq 0; h = 1,2,\ldots,k\right].
$$

For a fixed value of $y_k; 0 \leq y_k \leq d_k$.

$$
f(k,d_k) = f_k(y_k) + \min\left[\sum_{h=1}^{k-1} f_h(y_h) \mid \sum_{h=1}^{k-1} y_h = d_k - y_k, \text{and } y_h \geq 0; h = 1,2,\ldots,k-1\right].
$$

Using the Bellman's principle of optimality, we get the recurrence relation of the Dynamic Programming as

$$f(k,d_k)=\min_{0\le y_k\le d_k}[f_k(y_k)+f(k-1,d_k-y_k)],k\ge 2. \qquad (3.3.2)$$

For the first stage , that is for $k=1$

$$f(1,d_1)=f_1(d_1)\Rightarrow y_1^*=d_1 \qquad (3.3.3)$$

Where y_1^* is the optimum width of the first stratum . The relations (3.3.3) and (3.3.2) are solved recursively for each $k=1,2,...,L$ and $0\le d_k\le d$ and $f(L,d)$ is obtained. From $f(L,d)$ the optimum width of L^{th} stratum, y_L^* ,is obtained; from $f(L-1,d-y_L^*)$ the optimum width of $(L-1)^{th}$ stratum, y_{L-1}^* is obtained and so on until y_1^* is obtained.

In Chapter IV of this dissertation examples of the application of Dynamic Programming Technique for various frequency distributions of the stratification variable are discussed.

CHAPTER-IV

APPLICATION OF DYNAMIC PROGRAMMING TECHNIQUE WHEN THE STRATIFICATION VARIABLE FOLLOWS SOME SPECIALIZED DISTRIBUTION

4.1 INTRODUCTION:

In chapter III the problem of determining the OSB is formulated as an MPP that can be solved using Dynamic Programming Technique. The general formulation of the problem is given in (3.2.11). For different distribution $f()$ in (3.2.11) takes on different forms.

In this chapter Dynamic Programming Technique is applied for determining OSB through OSW for the following distribution.

(i) Rectangular

(ii) Right Triangular

(iii) Exponential

(iv) Log- normal

The following research papers were consulted in the preparation this

chapter. Khan et al (2002), Khan et al (2005), and unpublished thesis of A. H. Ansari (2008)

4.2 OSB WHEN THE STUDY VARIABLE HAS A RECTANGULAR DISTRIBUTION:

Let X follow a Rectangular (Uniform) Distribution in the interval $[a,b]$. Then

$$f(x)=\frac{1}{b-a}; \qquad a \le x \le b$$

$$=0; \qquad \text{otherwise.}$$

Using equations

$$W_h = \int_{x_{h-1}}^{x_h} f(x)dx \tag{4.2.1}$$

$$\sigma_h^2 = \frac{1}{W_h}\int_{x_{h-1}}^{x_h} x^2 f(x)dx - \mu_h^2 \tag{4.2.2}$$

where

$$\mu_h = \frac{1}{W_h}\int_{x_{h-1}}^{x_h} xf(x)dx \tag{4.2.3}$$

it can be verified that

$$W_h = \frac{y_h}{b-a}$$

$$\mu_h = \frac{y_h + 2x_{h-1}}{2}$$

and

$$\sigma_h^2 = \frac{y_h^2}{12}$$

Using the above values of W_h, μ_h and σ_h^2 the MPP

$$\text{Minimize} \quad \sum_{h=1}^{L} f_h(y_h)$$

$$\text{subject to} \quad \sum_{h=1}^{L} y_h = d$$

$$\text{and} \quad y_{h;} \geq 0 \qquad h = 1,2,...,L. \qquad (4.2.4)$$

Can be expressed as

$$\text{Minimize} \quad \sum_{h=1}^{L} \frac{y_h^2}{2\sqrt{3}(b-a)}$$

$$\text{subject to} \cdot \quad \sum_{h=1}^{L} y_h = d$$

$$\text{and} \quad y_h \geq 0; \qquad h = 1,2,...,L. \qquad (4.2.5)$$

where $d = b - a$

To illustrate the computational procedure we take $[a,b] = [1,2]$ and

$L = 6$,

which gives MPP (4.2.5) as:

$$\text{Minimize} \quad \sum_{h=1}^{6} \frac{y_h^2}{2\sqrt{3}}$$

$$\text{subject to} \quad \sum_{h=1}^{6} y_h = 1,$$

$$\text{and} \quad y_{h;} \geq 0 \qquad h = 1,2,...,6. \qquad (4.2.6)$$

For the first stage $(K = 1)$

$$f(1,d_1) = \frac{d_1^2}{2\sqrt{3}} \qquad \text{at} \qquad y_1^* = d_1. \qquad (4.2.7)$$

For second stage

$$f(2,d_2) = \min_{0 \leq y_2 \leq d_2} \left[\frac{y_2^2}{2\sqrt{3}} + f(1,d_2 - y_2) \right]$$

$$= \min_{0 \leq y_2 \leq d_2} \left[\frac{y_2^2}{2\sqrt{3}} + \frac{(d_2 - y_2)^2}{2\sqrt{3}} \right].$$

Using differential calculus for minimization we get

$$f(2,d_2)=\frac{d_2^2}{4\sqrt{3}} \quad \text{at} \quad y_2^*=\frac{d_2}{2}. \tag{4.2.8}$$

For third stage $(K=3)$

$$f(3,d_3)=\min_{0\le y_3\le d_3}\left[\frac{y_3^2}{2\sqrt{3}}+f(2,d_3-y_3)\right]$$

$$=\min_{0\le y_2\le d_2}\left[\frac{y_3^2}{2\sqrt{3}}+\frac{(d_3-y_3)^2}{4\sqrt{3}}\right].$$

Using differential calculus for minimization we get

$$f(3,d_3)=\frac{d_3^2}{6\sqrt{3}} \quad \text{at} \quad y_3^*=\frac{d_3}{3}. \tag{4.2.9}$$

Similarly for the fourth and fifth stages we get

$$f(4,d_4)=\frac{d_4^2}{8\sqrt{3}} \quad \text{at} \quad y_4^*=\frac{d_4}{4} \tag{4.2.10}$$

$$f(5,d_5)=\frac{d_5^2}{10\sqrt{3}} \quad \text{at} \quad y_5^*=\frac{d_5}{5}. \tag{4.2.11}$$

For the final stage $(K=6)$

$$f(6,d_6)=\min_{0\le y_6\le d_6}\left[\frac{y_6^2}{2\sqrt{3}}+\frac{(d_6-y_6)^2}{10\sqrt{3}}\right]$$

$$f(6,1) = \min_{0 \le y_6 \le 1} \left[\frac{y_6^2}{2\sqrt{3}} + \frac{(1-y_6)^2}{10\sqrt{3}} \right]$$

$$\Rightarrow f(6,1) = \frac{1}{12\sqrt{3}} = 0.048112522 \text{ at } y_6^* = \frac{1}{6} = 0.166667. \qquad (4.2.12)$$

From (4.2.12), $d_5 = d_6 - y_6^* = 1 - 0.66667 = 0.833333.$

Substituting this value of d_5 in (4.2.11)

$$y_5^* = \frac{0.833333}{5} = 0.166666.$$

Proceeding in this manner we get

$$y_4^* = 0.166667,\ y_3^* = 0.166666,\ y_2^* = 0.166667 \text{and } y_1^* = 0.166667.$$

The optimum strata boundaries are then obtained as

$$x_1^* = x_0 + y_1^* = 1 + 0.166667 = 1.166667$$

$$x_2^* = x_1^* + y_2^* = 1.166667 + 0.166667 = 1.333334$$

$$x_3^* = x_2^* + y_3^* = 1.333334 + 0.166666 = 1.500000$$

$$x_4^* = x_3^* + y_4^* = 1.500000 + 0.166667 = 1.666667$$

$$x_5^* = x_4^* + y_5^* = 1.666667 + 0.166666 = 1.833333$$

with the optimum value of the objective function $\sum_{h=1}^{6} f_h(y_h)$ as

$f(6,1) = 0.048112522.$

4.3 OSB WHEN THE STUDY VARIABLE FOLLOWS A RIGHT- TRIANGULAR DISTRIBUTION:

Let x follow the Right Triangular distribution in the interval $[a,b]$.

In this case we have

$$f(x) = \frac{2(b-x)}{(b-a)^2}; \qquad a \le x \le b$$

$$= 0; \qquad \text{otherwise}$$

using (4.2.1), (4.2.2) and (4.2.3) W_h, μ_h and σ_h^2 are obtained as

$$W_h = \frac{y_h(2a_h - y_h)}{(b-a)^2}$$

$$\mu_h = \frac{3b(y_h + 2x_{h-1}) - 2\left(y_h^2 + 3x_{h-1}y_h + 3x_{h-1}\right)}{3(2a_h - y_h)}$$

and

$$\sigma_h^2 = \frac{y_h^2\left(y_h^2 - 6a_h y_h + 6a_h^2\right)}{18(2a_h - y_h)^2}$$

where $a_h = 1 - x_h;\ h = 1,2,...,6.$

using the above values of W_h, μ_h and σ_h^2 the MPP (4.2.4) can be expressed as

$$\text{Minimize} \quad \sum_{h=1}^{L} \frac{y_h^2 \sqrt{y_h^2 - 6a_h y_h + 6a_h^2}}{3\sqrt{2}(b-a)^2}$$

$$\text{subject to} \quad \sum_{h=1}^{L} y_h = d,$$

$$\text{and} \quad y_{h \geq 0;} \quad h = 1,2,...,L. \qquad (4.3.1)$$

where $d = b - 1.$

To illustrate the computational procedure we take $[a,b] = [0,1]$ and $L = 6,$ which gives MPP (4.3.1) as:

$$\text{Minimize} \quad \sum_{h=1}^{6} \frac{y_h^2 \sqrt{y_h^2 - 6a_h y_h + 6a_h^2}}{3\sqrt{2}}$$

$$\text{Subject to} \quad \sum_{h=1}^{6} y_h = 1,$$

$$\text{and} \quad y_h \geq 0 \qquad h = 1,2,...,6. \qquad (4.3.2)$$

using the recurrence relations

$$f(k,d_k)=\min_{0\le y_k\le d_k}[f_k(y_k)+f(k-1,d_k-y_k)],k\ge 2.$$

For the first stage, that is for $k=1$

$$f(1,d_1)=f_1(d_1)\Rightarrow y_1^*=d_1$$

For solving MPP (4.3.2) we get:

For the first stage $(k=1)$

$$f(1,d_1)=\frac{d_1^2\sqrt{d_1^2-6d_1+6}}{3\sqrt{2}} \quad \text{at} \quad y_1^*=d_1 \qquad (4.3.3)$$

and for stages $k\ge 2$

$$f(k,d_k)=\min_{0\le y_k\le d_k}\left[\frac{y_k^2\sqrt{y_k^2-6a_ky_k+6a_k^2}}{3\sqrt{2}}+f(k-1,d_k-y_k)\right] \qquad (4.3.4)$$

where $a_k=1-x_{k-1}=1-(x_0+y_1+y_2+...+y_{k-1})$

$=1-(y_1+y_2+...y_{k-1})=1-d_{k-1}\Rightarrow a_k=1-(d_k-y_k)=1-d_k+y_k.$

substituting this value of a_k in (4.3.4) and executing the computer program developed for the solution procedure given in chapter –iii the optimum strata widths are obtained as:

$y_1^* = 0.112647, \quad y_2^* = 0.120353, \quad y_3^* = 0.130930,$

$y_4^* = 0.146071, \; y_5^* = 0.173603$ and $y_6^* = 0.316396$

with the optimum value of the objective function $\sum_{h=1}^{6} f_h(y_h)$ as

, $f(6,1) = 0.0420973209$, which gives the OSB as:

$$x_1^* = x_0 + y_1^* = 0 + 0.112647 = 0.112647$$

$$x_2^* = x_1^* + y_2^* = 0.112647 + 0.120353 = 0.233000$$

$$x_3^* = x_2^* + y_3^* = 0.233000 + 0.130930 = 0.363930$$

$$x_4^* = x_3^* + y_4^* = 0.363930 + 0.146071 = 0.510001$$

$$x_5^* = x_4^* + y_5^* = 0.510001 + 0.173603 = 0.683604$$

4.4 OSB WHEN THE STUDY VARIABLE FOLLOW AN EXPONENTIAL DISTRIBUTION:

Let the stratification variable X follows the exponential distribution with parameter $\lambda = 0$, that is

$$f(x) = \frac{1}{\lambda} e^{-x/\lambda}, \qquad 0 \le x \le \infty$$

$$= 0, \qquad \text{otherwise}$$

In practice the actual population are often finite, so assuming the largest value of x in the population as D, the above frequency function can be approximated as

$$f(x)=\frac{1}{\lambda}e^{-x/\lambda}, \qquad 0\le x\le D \qquad (4.4.1)$$

$$=0, \qquad \text{elsewhere}$$

Note that we have here $x_0=0$ and $x_L=D$. If D is sufficiently large, (4.4.1) can be considered as an approximate exponential density otherwise the truncated exponential density is to be used and the expressions (4.4.2) – (4.4.4) are to be worked out accordingly.

Using equations

$$W_h=\int_{x_{h-1}}^{x_h} f(x)\,dx$$

$$\sigma_h^2=\frac{1}{W_h}\int_{x_{h-1}}^{x_h} x^2 f(x)dx-\mu_h^2$$

$$\text{where } \mu_h=\frac{1}{W_h}\int_{x_{h-1}}^{x_h} x f(x)dx$$

For exponential distribution

$$W_h = e^{-x_{h-1}/\lambda}\left(1-e^{-y_h/\lambda}\right) \tag{4.4.2}$$

$$\mu_h = \frac{(\lambda + x_{h-1})\left(1-e^{-y_h/\lambda}\right) - y_h e^{-y_h/\lambda}}{1-e^{-y_h/\lambda}} \tag{4.4.3}$$

and

$$\sigma_h^2 = \frac{\lambda^2\left(1-e^{-y_h/\lambda}\right)^2 - y_h^2 e^{-y_h/\lambda}}{\left(1-e^{-y_h/\lambda}\right)^2} \tag{4.4.4}$$

Using (4.4.2), (4.4.3) and (4.4.4), the problem of determining optimum strata boundaries, when the frequency of the main study variable X is given by (4.4.1), may be expressed as

$$\text{Minimize} \quad \sum_{h=1}^{L} e^{-x_{h-1}/\lambda}\sqrt{\lambda^2\left(1-e^{-y_h/\lambda}\right)^2 - y_h^2 e^{-y_h/\lambda}}$$

$$\text{subject to} \quad \sum_{h=1}^{L} y_h = d \tag{4.4.5}$$

$$\text{and} \quad y_h \geq 0;\ h = 1,2,...,L$$

where d is obtained by equation $x_L - x_0 = d$ with $x_0 = 0$ and $x_L = D$.

Consider the following sub problem for first $k(< L)$ strata.

$$\text{Minimize} \quad \sum_{h=1}^{k} f_h(y_h)$$

$$\text{Subject to} \quad \sum_{h=1}^{k} y_h = d_k \tag{4.4.6}$$

$$\text{and} \qquad y_h \geq 0;\ h = 1,2,...,k$$

Where $d_k < d$ is the total width available for division into k strata.

Note that $d_k = d$ for $k = L$

Also $d_k = y_1 + y_2 + ... + y_k$

$$d_{k-1} = y_1 + y_2 + ... + y_{k-1} = d_k + y_k$$

$$d_{k-2} = y_1 + y_2 + ...y_{k-2} = d_{k-1} - y_{k-1}$$

. .

. .

. .

$$d_2 = y_1 + y_2 = d_3 - y_3 \qquad \text{and} \qquad d_1 = y_1 = d_2 - y_2$$

If $f(k,d_k)$ denotes the minimum value of the objective function of (4.4.6), then

$$f(k,d_k) = \min\left[\sum_{h=1}^{k} f_h(y_h) / \sum_{h=1}^{k} y_h = d_k \text{ and } y_h \geq 0; h = 1,2,...k\right]$$

with the above definition of $f(k,d_k)$ the recurrence relations of the dynamic programming takes the form

$$f(k,d_k)=\min_{0\le y_k\le d_k}\left(f_k(y_k)+f(k-1,d_k-y_k)\right),k\ge 2 \qquad (4.4.7)$$

For the first stage $(i.e.k=1)$

$$f(1,d_1)=f_1(d_1)\Rightarrow y_1=d_1 \qquad (4.4.8)$$

From $f(L,d)$ the optimum width of L^{th} stratum, y_L, is obtained from $f(L-1,d-y_L)$ the optimum width of $(L-1)^{th}$ stratum, y_{L-1}, is obtained and so on until y_1 is obtained.

Using (4.4.8) and (4.4.7) the recurrence relations for MPP (4.4.5) are as given

For first stage $(k=1)$

$$f(1,d_1)=\sqrt{\lambda^2\left(1-e^{-d_1/\lambda}\right)^2-d_1^2e^{-d_1/\lambda}}$$

$$\text{at} \qquad y_1=d_1 \qquad (4.4.9)$$

because $x_{k-1}=x_0=0$, when $k=1$.

for the stage k, where $k\ge 2$

$$f(k,d_k)=\min_{0\le y_k\le d_k}\left[e^{(-d_k-y_k)/\lambda}\sqrt{\lambda^2\left(1-e^{-y_k/\lambda}\right)^2-y_k^2e^{-y_k/\lambda}}\right.$$

$$\left.+f(k-1,d_k-y_k)\right] \qquad (4.4.10)$$

because $x_{k-1}=x_0+y_1+...+y_{k-1}=d_k-y_k$

4.5 A NUMERICAL EXAMPLE:

Executing the computer program in "JAVA-SDK2" given in the section (4.6) the recurrence relations (4.4.9) and (4.4.10) are solved to find optimum stratum widths $y_k;(k=1,2,...,L)$ for the exponential study variable with density function given in (4.4.1), with $D=20$ and $\lambda=1$.

Table 1 gives the optimum values of y_h, x_h, and $\sum_{h=1}^{L}f_h(y_h)$ for $L=2,3,4$ and 5.

Table-1

No. of strata L	Strata widths y_h^*	Strata boundary points $x_h^* = x_{h-1}^* + y_h^*$	Optimum values of the objective function $\sum_{h=1}^{L} f_h(y_h) = \sum_{h=1}^{L} W_h \sigma_h$
2	$y_1^* = 1.2610$ $y_2^* = 18.7390$	$x_1^* = x_0 + y_1^* = 1.2610$	0.5341
3	$y_1^* = 0.7678$ $y_2^* = 1.2501$ $y_3^* = 17.9821$	$x_1^* = x_0 + y_1^* = 0.7678$ $x_2^* = x_1^* + y_2^* = 2.0179$	0.3648
4	$y_1^* = 0.5509$ $y_2^* = 0.7638$ $y_3^* = 1.2513$ $y_4^* = 17.4340$	$x_1^* = x_0 + y_1^* = 0.5509$ $x_2^* = x_1^* + y_2^* = 1.3147$ $x_3^* = x_2^* + y_3^* = 2.5650$	0.2770
5	$y_1^* = 0.4393$ $y_2^* = 0.5610$ $y_3^* = 0.7569$ $y_4^* = 1.2688$ $y_5^* = 16.9740$	$x_1^* = x_0^* + y_1^* = 0.4393$ $x_2^* = x_1^* + y_2^* = 1.0003$ $x_3^* = x_2^* + y_3^* = 1.7572$ $x_4^* = x_3^* + y_4^* = 2.0260$	0.2233

The total width available for cutting stratum boundaries is taken as 20 units, i.e the largest population value $x_L = D = 20$, because the area to the right of $x = 20$ for exponential distribution is almost zero,

when $\lambda = 1$.

4.6 THE COMPUTER PROGRAM FOR EXPONENTIAL DISTRIBUTION (IN JAVA – SDK2):

```
import java.io.*;
import java.util.*;

public class OptimumNew
{

        private RandomAccessFile randReader[] = null;
        private double e=2.718281828;
        private double increment = 0.10;
        private int intPreci = 1;
        private int intStage = 1;
        private int Dk = 999;
        DataOutputStream outputStream[];
        double storedFk[];

        public static void main(String args[])
        {
                new OptimumNew();
        }

        public OptimumNew()
        {
                System.out.println("enter the Stage value (1 to 9 only):");
                String str = Readline.readLine();
                intStage = Integer.parseInt(str);


                System.out.println("enter the summation Yk ( Dk ) value (integer
only):");
                str = Readline.readLine();
                Dk = Integer.parseInt(str);
                System.out.println("enter the desired precesion 1- 9 (integer
only):");
                str = Readline.readLine();
                intPreci = Integer.parseInt(str);

                try
                {
                        randReader = new RandomAccessFile[intStage];
```

```
                    for(int i =0; i < intStage; i++)
                    {
                            File file = new File("./Stage"+(i+1)+".txt");
                            randReader[i] = new RandomAccessFile(file, "r");
                    }
                    FileOutputStream fos[] = new
 FileOutputStream[intStage];
                     outputStream = new DataOutputStream[intStage];
                    for(int i =0; i < intStage; i++)
                    {
                            File file = new File("./Stage"+(i+1)+".txt");
                            fos[i] = new FileOutputStream(file);
                            outputStream[i] = new DataOutputStream(fos[i]);
                    }
                    funF1D1()   ;
                    for(int i = 1; i < intStage; i++)
                            funFkDk(i)   ;
                    backWardCalculation();

            }
            catch(Exception ex)
            {
                    ex.printStackTrace();

            }
    }


            void funF1D1()
            {
                    storedFk = new double[(int)(Dk*Math.pow(10,
intPreci)+1)];
                    double Y1=0;
                    double dblTmp1 = 0;
                    double fx= 0;
                    double d1 = 0;
                    long d1Count=0;
                    int count = 0;
                    String strD1 = "", strFx="", strY1="";
                    increment = Math.pow(10, -intPreci);
                    //System.out.println(increment);
                    while(d1 <= Dk)
                    {
```

```
                        Y1 = d1;
                        fx = (1 - Math.pow(e, -Y1))*(1 - Math.pow(e, -
Y1)) - Y1*Y1*Math.pow(e, -Y1);
                        if(fx<0.0)
                        {
                                System.out.println("SQRT OF THE -VE
QUANTITY in funFkDk_");
                                System.out.println("d1="+d1+",
Y1="+Y1+" , fx= "+fx+"\n");
                                System.exit(0);
                        }
                        else
                                dblTmp1=Math.sqrt(fx);

                        fx=Math.pow(e, -(d1-Y1))*dblTmp1;
                        storedFk[count] = fx;
                        count++;
                        strFx = Double.toString(fx);
                        while(strFx.length() < 25)
                        {
                                strFx = "0" + strFx;
                        }
                        strY1 = Double.toString(Y1);
                        while(strY1.length() < 25)
                        {
                                strY1 = "0" + strY1;
                        }
                        strD1 = Double.toString(d1);
                        while(strD1.length() < 25)
                        {
                                strD1 = "0" + strD1;
                        }
                        try
                        {
                                outputStream[0].writeBytes(strD1+" " +
strY1+" " + strFx+"\n");
                        }
                        catch(Exception ex)
                        {
                                ex.printStackTrace();
                        }
                        //d1 += increment;
                        d1Count++;
```

```
                        d1 = d1Count*Math.pow(10, -
intPreci);//increment;
                    }

                }

                double readFkDk1(int K, double Dk)
                {
                    double tmpDk = Dk*Math.pow(10, intPreci);
                    long lDk = (long)Dk;
                    long n1 = (long)tmpDk;
        //Math.round(Dk*Math.pow(10, intPreci));
                    String str= "";
                    double ret=0, data1 =0, data2 =0;
                    n1 = n1*78;
                    try
                    {
                        if(n1 < 0 || n1 > randReader[K].length()) return 0;
                        randReader[K].seek(n1);
                        str = randReader[K].readLine();
                        data1 = Double.parseDouble(str.substring(51));
                        if(str != null && str.length() >= 75)
                        {
                            data2 =
Double.parseDouble(str.substring(26, 51));
                        }
                        else
                         data2 = data1;
                         ret = data1+ (data2-data1)*(Dk*100 -
lDk*100)/100;
                        //System.out.println( "fkdk- Dk passed =" + Dk + ",
line = "+n1/78 +", Fx=" + ret );
                    }
                    catch(Exception ex)
                    {
                        System.out.println( "fkdk- Dk passed =" + Dk + ",
line = "+n1/78 +", str=" + str );
                        ex.printStackTrace();
                        System.exit(0);

                    }
                    return ret;
                }
```

```
double readFkDk(int K, double Dk)
{
        double tmpDk = Dk*Math.pow(10, intPreci);
        long lDk = (long)Dk;
        int n1 = (int)tmpDk; //Math.round(Dk*Math.pow(10,
intPreci));
        String str= "";
        double ret=0, data1 =0, data2 =0;
        try
        {
                data1 = storedFk[n1];
                if(n1 < storedFk.length-1)
                {
                        data2 = storedFk[n1+1];
                }
                else
                 data2 = data1;
                 ret = data1+ (data2-data1)*(Dk*100 -
lDk*100)/100;
                //System.out.println( "fkdk- Dk passed =" + Dk + ",
line = "+n1/78 +", Fx=" + ret );
        }
        catch(Exception ex)
        {
                System.out.println( "readFkDk- Dk passed =" + Dk
+ ", line = "+n1/78 +", str=" + str );
                ex.printStackTrace();
                System.exit(0);

        }
        return ret;
}

void funFkDk(int K)
{
        if(K > 1)
                readStoredFk(K);
        double Yk=0;
        double dblTmp1 = 0;
        long multi = (long)Math.pow(10, intPreci+1);
        double fx= 0;
```

```
double dk = increment;
long dkCount =1;
String strD1 = "", strFx="", strY1="";
//for 0<= Yk<= Dk
double increL = increment/10;
double minFx =999999;
double minYk = 45667;
try
{

    outputStream[K].writeBytes("00000000000000000000000000
00000000000000000000000000 00000000000000000000000000\n");
}
catch(Exception ex)
{
        ex.printStackTrace();
}
double lowLimit = 0, upperLimit = 0, increTmp =0;
while(dk <= Dk)
{
        //find min
        minFx = 999999;
        minYk = 999999;
        lowLimit = 0;
        upperLimit = dk;
        if(increL <= .01) increTmp = .1;
        if(upperLimit >= 20) increTmp = 1;
        if(upperLimit > 200) increTmp = 10;
        if(upperLimit <= 20*increL) increTmp = increL;
        while(increL <= increTmp)
        {
                minFx = 999999;
                Yk = lowLimit;
                //while(Yk<=dk)
                int count =0;
                while(Yk<=upperLimit)
                {
                        count++;
                        //calculateFkDk();
                        dblTmp1 = (1 - Math.pow(e, -
Yk))*(1 - Math.pow(e, -Yk)) - Yk*Yk*Math.pow(e, -Yk);
                        if(dblTmp1 < 0.0)
                        {
```

```
                                        System.out.println("Sqrt of the
-ve quantity in fun2_");

        System.out.println("Yk="+Yk+" ,increTmp="+increTmp+" ,dk="+dk+"
,dblTmp1="+dblTmp1+" \n"+Math.pow(e, -Yk));
                                        System.exit(0);
                                }
                                else
                                        dblTmp1 =
Math.sqrt(dblTmp1);

                                fx = Math.pow(e,-(dk-Yk)) *
dblTmp1 + readFkDk(K-1, dk-Yk);
                                if(minFx > fx)
                                {
                                        minFx = fx;
                                        minYk = Yk;
                                }
                                Yk += increTmp;
                        }
                        lowLimit = minYk-increTmp;
                        upperLimit = minYk + increTmp;
                        if(upperLimit > dk ) upperLimit = dk;
                        if(lowLimit < 0 ) lowLimit = 0;
                        increTmp = increTmp/10;
                }

                strFx = Double.toString(minFx);
                while(strFx.length() < 25)
                {
                        strFx = "0" + strFx;
                }
                strY1 = Double.toString(minYk);
                while(strY1.length() < 25)
                {
                        strY1 = "0" + strY1;
                }
                strD1 = Double.toString(dk);
                while(strD1.length() < 25)
                {
                        strD1 = "0" + strD1;
                }
                try
```

```
                        {
                                outputStream[K].writeBytes(strD1+" " +
strY1+" " + strFx+"\n");
                        }
                        catch(Exception ex)
                        {
                                ex.printStackTrace();
                        }
                        Yk = dk;
                        dkCount++;
                        dk = dkCount*Math.pow(10, -
intPreci);//increment;
                }
                try
                {
                        System.out.println(K+" file-
"+randReader[K].length());
                }
                catch(Exception ex)
                {
                        ex.printStackTrace();
                }
        }

        void backWardCalculation()
        {
                try
                {
                File tmpFile = new File("./resultNew.txt");
                RandomAccessFile rand = new
RandomAccessFile(tmpFile, "rw");
                rand.seek(rand.length());
                double fxx[] = new double[intStage];
                double fyy[] = new double[intStage];
                double fdd[] = new double[intStage];
                int kk = intStage-1;
                fxx[kk] = readFkDk1(kk, Dk);
                fyy[kk] = readYk(kk, Dk);
                fdd[kk]= Dk;
                rand.writeBytes("\n Date: " + new Date() + "\nNumber of
stage = "+ intStage +", Dk = " + Dk +  ", Precision = "+ intPreci
                );
                //System.out.println( "Yk- Dk  =" + Dk );
```

```
                    for( int i =kk-1; i >= 0 ; i--)
                    {
                            fdd[i]= fdd[i+1] - fyy[i+1];
                            fxx[i] = readFkDk1(i, fdd[i+1]-fyy[i+1]);
                            fyy[i] = readYk(i, fdd[i+1]-fyy[i+1]);
                            //System.out.println(fdd[i+1] + ", fdd=" + fdd[i] );
                    }

                    for( int i =0; i <= kk ; i++)
                    {
                            rand.writeBytes("\nY" + (i+1) + " = " + fyy[i] +",
D" + (i+1) + " = " + fdd[i]);
                    }
                    rand.writeBytes("\nfx" + (kk+1) + " = " + fxx[kk]+
"\n\n\n\n");
            }
            catch(Exception ex)
            {
                    ex.printStackTrace();
            }
            }

                    double readYk(int K, double Dk)
                    {
                            double tmpDk = Dk*Math.pow(10,
intPreci);
                            long lDk = (long)Dk;
                            long n1 = (long)tmpDk;
      //Math.round(Dk*Math.pow(10, intPreci));
                            double ret=0, data1 =0, data2 =0;
                            String str= "";
                            n1 = n1*78;
                            try
                            {
                                    if(n1 < 0 || n1 >
randReader[K].length()) return 0;
                                    randReader[K].seek(n1);
                                    str= randReader[K].readLine();
                                    data1 =
Double.parseDouble(str.substring(26,51));
                                    str= randReader[K].readLine();
                                    if(str != null && str.length() >= 75)
                                    {
```

```
                                        data2 =
Double.parseDouble(str.substring(26, 51));
                                    }
                                    else
                                     data2 = data1;
                                     ret = data1+ (data2-data1)*(Dk*100
-lDk*100)/100;
                                    //System.out.println( "Dk passed =" +
Dk + ", line = "+n1/78 +", Fx=" + ret );
                                }
                                catch(Exception ex)
                                {
                                    System.out.println( K+",Dk passed
=" + Dk + ", line = "+n1/78 +", str=" + str );
                                    ex.printStackTrace();

                                }
                                return ret;
                        }


                        void readStoredFk(int k)
                        {
                                k--;
                                try
                                {
                                        File file = new File("./Stage"+(k+1)+".txt");
                                        RandomAccessFile randTmp = new
RandomAccessFile(file, "r");
                                //      randReader[k].seek(0);
                                        System.out.println( "filelength read= "
+randReader[k].length() );
                                        String str = null;
                                        int line = 0;
                                        System.out.println( "filelength= "
+randTmp.length() + ", array=" + storedFk.length);

                                        while((str = randTmp.readLine()) != null
&& line < storedFk.length)
                                        {
                                            storedFk[line] =
Double.parseDouble(str.substring(51));
                                            line++;
```

```
                                        }
                                        System.out.println( "k= " +k + ", line=" +
line);
                                }
                                catch(Exception ex)
                                {
                                        ex.printStackTrace();
                                }

                        }

//*****************************************************************
*****
        static class Readline
        {

                public static void main(String args[])
                {
                        try{
                                // 1. Create an InputStreamReader using the
standard input stream
                                InputStreamReader isr = new InputStreamReader(
System.in );

                                // 2. Create a BufferedReader using the
InputStreamReader created.
                                BufferedReader stdin = new BufferedReader( isr );

                                // 3. Don't forget to prompt the user
                                System.out.print( "Type some data for the program:
" );

                                // 4. Use the BufferedReader to read a line of text
from the user.
                                String input = stdin.readLine();

                                // 5. Now, you can do anything with the input string
that you need to.
                                // Like, output it to the user.
                                System.out.println( "input = " + input );
                        }catch(Exception ex){ex.printStackTrace();}
                }
```

```
        public static String readLine()
        {
                String input = "0";
                try{
                        // 1. Create an InputStreamReader using the
standard input stream
                        InputStreamReader isr = new InputStreamReader(
System.in );

                        // 2. Create a BufferedReader using the
InputStreamReader created.
                        BufferedReader stdin = new BufferedReader( isr );

                        // 4. Use the BufferedReader to read a line of text
from the user.
                        input = stdin.readLine();


                }catch(Exception
ex){ex.printStackTrace();System.exit(0);}
                finally
                {

                }
                        return input;
        }
    }


}
```

4.7 OSB WHEN THE STUDY VARIABLE HAS A LOG-NORMAL DISTRIBUTION:

The Log-normal distribution is a positively skewed distribution, meaning that most of the distribution is concentrated around the left end, closest to zero. Surveyors may use the Log-normal distribution for a positive valued study variable that might increase without limits, such as the value of securities (financial applications) or properties (real estate applications) or the failure rate of electronic parts (engineering applications).

A variable X is Log-normally distributed if $Y = \ln(X)$ is normally distributed where "ln" stands for the natural logarithm. The general formula for the probability density function of the Log-normal distribution is

$$f(x) = \frac{\exp\left[-\left(\left(\ln((x-\theta)/m)\right)^2/\left(2\sigma^2\right)\right)\right]}{(x-\theta)\sigma\sqrt{2\pi}}; \qquad x > \theta, m > 0, \sigma > 0, \quad (4.7.1)$$

where σ is the shape parameter, θ is the location parameter and m is the scale parameter. With $\theta = 0$ and $m = 1$ (4.5.1) gives the standard Log-normal density as

$$f(x) = \frac{\exp\left[-\left(\left(\ln x\right)^2/\left(2\sigma^2\right)\right)\right]}{x\sigma\sqrt{2\pi}}; \quad x > 0, \sigma > 0. \qquad (4.7.2)$$

Using the definitions (3.2.5),(3.2.7), and (3.2.6) of W_h, μ_h and σ_h^2, it can be seen that

$$W_h = \frac{1}{2}\left(erf\left(\frac{\ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right), \qquad (4.7.3)$$

$$\mu_h = -\exp\left(\frac{\sigma^2}{2} \right) \frac{\left(erf\left(\frac{\sigma^2 - \ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\sigma^2 - \ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right)}{\left(erf\left(\frac{\ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right)},$$

(4.7.4)

$$\sigma_h^2 = \frac{1}{\left[\frac{1}{2}\left(erf\left(\frac{\ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right]^2}$$

$$\left\{ \left[-\frac{1}{2}\exp(2\sigma^2) \left(erf\left(\frac{2\sigma^2 - \ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{2\sigma^2 - \ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right] \right.$$

$$\left[\frac{1}{2}\left(erf\left(\frac{\ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right]$$

$$\left. - \left[\frac{1}{2}\exp\left(\frac{\sigma^2}{2} \right) \left(erf\left(\frac{\sigma^2 - \ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\sigma^2 - \ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right]^2 \right\}$$

(4.7.5)

Note that an error function is used to counter the integrations with Log-normal density function. The probability that a Log-normal variate assumes a value in the range $[z_1, z_2]$ is given by:

$$\frac{1}{\sqrt{2\pi}} \int_{z_1}^{z_2} \exp\left(\frac{-x^2}{2}\right) dx = \frac{1}{2}[erf(z_2) - erf(z_1)]. \qquad (4.7.6)$$

Common properties of the error function include:

$$erf(-z) = -erf(z),\ erf(0) = 0,\ erf(\infty) = 1, \qquad erf(-\infty) = -1$$

Using (4.7.3) and (4.7.5) the MPP

$$\text{Minimize } \sum_{h=1}^{L} f_h(y_h),$$

$$\text{subject to } \sum_{h=1}^{L} y_h = d,$$

$$\text{and} \qquad y_h \geq 0; h = 1,2,...L.$$

may be expressed as:

$$\min imize \sum_{h=1}^{L} \left\{ \begin{array}{l} Sqrt\left\{ \left[-\frac{1}{2}\exp\left(2\sigma^2\right)\left(erf\left(\frac{2\sigma^2 - \ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) \right. \right. \right. \\ \left. \left. \left. - erf\left(\frac{2\sigma^2 - \ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right] \\ \left[\frac{1}{2}\left(erf\left(\frac{\ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) - erf\left(\frac{\ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right] \\ -\left[\frac{1}{2}\exp\left(\frac{\sigma^2}{2} \right)\left(erf\left(\frac{\sigma^2 - \ln(y_h + x_{h-1})}{\sigma\sqrt{2}} \right) \right. \right. \\ \left. \left. - erf\left(\frac{\sigma^2 - \ln(x_{h-1})}{\sigma\sqrt{2}} \right) \right) \right]^2 \end{array} \right\},$$

$$\text{subject to} \qquad \sum_{h=1}^{L} y_h = d,$$

$$\text{and} \qquad y_h \geq 0; \qquad h = 1,2,...,L. \qquad (4.7.7)$$

Let X follows the standard Log-normal distribution in the interval $[0.00001,\ 13.00001]$, that is, $a = x_0 = 0.00001, b = x_L = 13.00001$ and $\sigma = 1$. This gives $d = x_L - x_0 = 13$. The MPP (4.7.7) can now be expressed as:

$$\text{Minimize} \quad \sum_{h=1}^{L} \left\{ \begin{array}{l} Squrt\left\{ \left[-\frac{1}{2}\exp(2)\left(er\,f\left(\frac{2-\ln(y_h + x_{h-1})}{\sqrt{2}} \right) \right.\right.\right. \\ \left.\left.\left. \quad - er\,f\left(\frac{2-\ln(x_{h-1})}{\sqrt{2}} \right) \right) \right] \\ \left[\frac{1}{2}\left(er\,f\left(\frac{\ln(y_h + x_{h-1})}{\sqrt{2}} \right) - er\,f\left(\frac{\ln(x_{h-1})}{\sqrt{2}} \right) \right) \right] \\ -\left[\frac{1}{2}\exp\left(\frac{1}{2} \right)\left(er\,f\left(\frac{1-\ln(y_h + x_{h-1})}{\sqrt{2}} \right) \right.\right. \\ \left.\left. \quad - er\,f\left(\frac{1-\ln(x_{h-1})}{\sqrt{2}} \right) \right) \right]^2 \end{array} \right\}$$

subject to $\sum_{h=1}^{L} y_h = 13,$

and $y_h \geq 0; \quad h = 1,2,...,L.$ (4.7.8)

Also
$$\begin{aligned} x_{k-1} &= x_0 + y_1 + y_2 + ... + y_{k-1} \\ &= .00001 + y_1 + y_2 + ... + y_{k-1} \\ &= d_{k-1} + 0.00001 \\ &= d_k - y_k + 0.000001. \end{aligned}$$

Substituting this value of x_{k-1} in (4.7.8) and using equations

$$f(1,d_1) = f_1(d_1) \Rightarrow y_1^* = d_1,$$

$$f(k,d_k) = \min_{0 \leq y_k \leq d_k} [f_k(y_k) + f(k-1, d_k - y_k)], \; k \geq 2.$$

The recurrence relations for solving MPP (4.7.8) are obtained as:

For first stage $(k=1)$:

$$f(1,d_1)=Squrt\left\{\left[-\frac{1}{2}\exp(2)\left(erf\left(\frac{2-\ln(d_1+0.00001)}{\sqrt{2}}\right)-erf\left(\sqrt{2}\right)\right)\right]\right.$$

$$\left[\frac{1}{2}\left(erf\left(\left(\frac{\ln(d_1+0.00001}{\sqrt{2}}\right)\right)\right)\right]$$

$$\left.-\left[\frac{1}{2}\exp\left(\frac{1}{2}\right)\left(erf\left(\frac{1-\ln(d_1+0.00001)}{\sqrt{2}}\right)-erf\left(\frac{1}{\sqrt{2}}\right)\right)\right]^2\right\} \qquad (4.7.9)$$

at $y_1=d_1$,

and for the stages

$k\geq 2$;

$$f(k,d_k)=\min_{0\leq y_k\leq d_k}\left\{\begin{array}{l} Sqrt\left[-\frac{1}{2}\exp(2)\left(erf\left(\frac{2-\ln(d_k+0.00001)}{\sqrt{2}}\right)\right.\right. \\ \left.\left.-erf\left(\frac{2-\ln(d_k-y_k+0.00001)}{\sqrt{2}}\right)\right)\right] \\ \left[\frac{1}{2}\left(erf\left(\frac{\ln(d_k+0.00001)}{\sqrt{2}}\right)-erf\left(\frac{\ln(d_k-y_k+0.00001}{\sqrt{2}}\right)\right)\right] \\ -\left[\frac{1}{2}\exp\left(\frac{1}{2}\right)\left(erf\left(\frac{1-\ln(d_k+0.00001)}{\sqrt{2}}\right)\right.\right. \\ \left.\left.\left.-erf\left(\frac{1-\ln(d_k-y_k+0.00001)}{\sqrt{2}}\right)\right)\right]^2\right\} \\ +f(k-1,d_k-y_k) \end{array}\right\} \qquad (4.7.10)$$

Solving the recursive equations (4.7.9) and (4.7.10) by executing a computer program in C++ developed for the solution procedure given in section 4.8,

the OSWs are obtained. The results of optimum strata widths y_h^* and hence the optimum strata boundaries $x_h^* = x_{h-1}^* + y_h^*$ along with the values of the objective function $\sum_{h=1}^{L} f_h(y_h)$ for $L = 2,3,4,5$ and 6 are presented in Table 2.

Table 2:OSW and OSB for standard Log-normal study variable.

No.of Strata L	Optimum Strata Widths y_h^*	Optimum Strata Boundaries $x_h^* = x_{h-1}^* + y_h^*$	Optimum Values of the objective function $\sum_{h=1}^{L} f_h(y_h) = \sum_{h=1}^{L} W_h\sigma_h$
2	$y_1^* = 2.23652$ $y_2^* = 10.76348$	$x_1^* = 2.23653$	$f(2,13) = 0.8569355124$
3	$y_1^* = 1.30859$ $y_2^* = 2.35085$ $y_3^* = 9.34056$	$x_1^* = 1.30860$ $x_2^* = 3.65945$	$f(3,13) = 0.5773613579$
4	$y_1^* = 0.95459$ $y_2^* = 1.25278$ $y_3^* = 2.53417$ $y_4^* = 8.25846$	$x_1^* = 0.95460$ $x_2^* = 2.20738$ $x_3^* = 4.74155$	$f(4,13) = 0.4358095763$
5	$y_1^* = 0.76589$ $y_2^* = 0.84332$ $y_3^* = 1.36367$ $y_4^* = 2.62141$ $y_5^* = 7.40571$	$x_1^* = 0.76590$ $x_2^* = 1.60922$ $x_3^* = 2.97289$ $x_4^* = 5.59430$	$f(5,13) = 0.3501356776$

6	$y_1^* = 0.64767$ $y_2^* = 0.63431$ $y_3^* = 0.90957$ $y_4^* = 1.44256$ $y_5^* = 2.65047$ $y_6^* = 6.71542$	$x_1^* = 0.64768$ $x_2^* = 1.28199$ $x_3^* = 2.19156$ $x_4^* = 3.63412$ $x_5^* = 6.28459$	$f(6,13) = 0.2926636591$

4.8 THE COMPUTER PROGRAM FOR LOGNORMAL DISTRIBUTION:

The computer Program in C++ for Lognormal Distribution:

```
#include "iostream.h"

#include "math.h"

#include "assert.h"

#include "conio.h"

#include "stdio.h"

//#include "erfunc.h"

typedef double Number;

//#include <math.h>

//#include<iostream.h>

Number erff(Number x);

double geterf(double x){

// Number erff(Number );
```

```
// Number erffc(Number );
// Number x;
// cout << "\n\n Enter x.\n";
// cout << "\n Wanna check? Note that erf(0) = 0, and erf(infnity) = 1, \n";
// cout << "\n erf(-x) = - erf(x), erfc(x) = 1 - erf(x), erfc(-x) = 2 - erfc(x) \n";
// cin >> x;
return erff(x);
}
/*************************************************************
********
Returns the error function
erf(x) = 2*(int_0^x e^{-t^2} dt)/sqrt(pi) .
C.A. Bertulani May/15/2000
*************************************************************
********/
Number erff(Number x)
{
Number gammp(Number a, Number x);
return x < 0.0 ? -gammp(0.5,x*x) : gammp(0.5,x*x);
}
```

```
/*************************************************************
********
Returns the complementary error function
erfc(x) = 1- erf(x)
= 2*(int_x^infinity e^{-t^2} dt)/sqrt(pi) .
C.A. Bertulani May/15/2000
*************************************************************
********/

Number erffc(Number x)
{
Number gammp(Number a, Number x);
Number gammq(Number a, Number x);
return x < 0.0 ? 1.0+gammp(0.5,x*x) : gammq(0.5,x*x);
}
/*************************************************************
********
Returns the imcomplete gamma function
P(a,x) = (int_0^x e^{-t} t^{a-1} dt)/Gamma(a) , (a > 0).
C.A. Bertulani May/15/2000
```

```
**********************************************************
********/
Number gammp(Number a, Number x)
{
void gcf(Number *gammcf, Number a, Number x, Number *gln);
void gser(Number *gamser, Number a, Number x, Number *gln);
Number gamser,gammcf,gln;
if (x < 0.0 || a <= 0.0) cerr<< "Invalid arguments in routine gammp";
if (x < (a+1.0)) {
gser(&gamser,a,x,&gln);
return gamser;
} else { /* Use the continued fraction representation */
gcf(&gammcf,a,x,&gln); /* and take its complement. */
return 1.0-gammcf;
}
}
/**********************************************************
********
Returns the imcomplete gamma function
Q(a,x) = 1-P(a,x)
= (int_x^infinity e^{-t} t^{a-1} dt)/Gamma(a) , (a > 0).
```

```
C.A. Bertulani May/15/2000
**********************************************************************
********/
Number gammq(Number a, Number x)
{
void gcf(Number *gammcf, Number a, Number x, Number *gln);
void gser(Number *gamser, Number a, Number x, Number *gln);
Number gamser,gammcf,gln;
if (x < 0.0 || a <= 0.0) cerr << "Invalid arguments in routine gammq";
if (x < (a+1.0)) { /* Use the series representation */
gser(&gamser,a,x,&gln);
return 1.0-gamser; /* and take its complement. */
} else { /* Use the continued fraction representation. */
gcf(&gammcf,a,x,&gln);
return gammcf;
}
}
/**********************************************************************
********
Returns the imcomplete gamma function P(a,x) evaluated by its series
representation as gamser.
```

```
Also returns ln(Gamma(a)) as gln.

C.A. Bertulani May/15/2000

**********************************************************

********/

#define ITMAX 100

#define EPS 3.0e-7


void gser(Number *gamser, Number a, Number x, Number *gln)

{

Number gamma_ln(Number xx);

int n;

Number sum,del,ap;

*gln=gamma_ln(a);

if (x <= 0.0) {

if (x < 0.0) cerr << "x less than 0 in routine gser";

*gamser=0.0;

return;

} else {

ap=a;

del=sum=1.0/a;

for (n=1;n<=ITMAX;n++) {
```

```
++ap;
del *= x/ap;
sum += del;
if (fabs(del) < fabs(sum)*EPS) {
*gamser=sum*exp(-x+a*log(x)-(*gln));
return;
}
}
cerr << "a too large, ITMAX too small in routine gser";
return;
}
}
#undef ITMAX
#undef EPS
/*************************************************************
********
```

Returns the imcomplete gamma function Q(a,x) evaluated by its

continued fraction representation as gammcf.

Also returns ln(Gamma(a)) as gln.

C.A. Bertulani May/15/2000

```
**********************************************************
********/
#define ITMAX 100 /* Maximum allowed number of iterations. */
#define EPS 3.0e-7 /* Relative accuracy */
#define FPMIN 1.0e-30 /* Number near the smallest representable */
/* floating point number. */
void gcf(Number *gammcf, Number a, Number x, Number *gln)
{
Number gamma_ln(Number xx);
int i;
Number an,b,c,d,del,h;
*gln=gamma_ln(a);
b=x+1.0-a; /* etup fr evaluating continued fracion by modified
Lent'z */
c=1.0/FPMIN; /* method with b_0 = 0. */
d=1.0/b;
h=d;
for (i=1;i<=ITMAX;i++) { /* Iterate to convergence. */
an = -i*(i-a);
b += 2.0;
d=an*d+b;
```

```
if (fabs(d) < FPMIN) d=FPMIN;
c=b+an/c;
if (fabs(c) < FPMIN) c=FPMIN;
d=1.0/d;
del=d*c;
h *= del;
if (fabs(del-1.0) < EPS) break;
}
if (i > ITMAX) cerr << "a too large, ITMAX too small in gcf";
*gammcf=exp(-x+a*log(x)-(*gln))*h; /* Put factors in front. */
}
#undef ITMAX
#undef EPS
#undef FPMIN
/*************************************************************
*******
Returns the value of ln[Gamma(xx)] for xx > 0
*************************************************************
*******/
Number gamma_ln(Number xx)
```

```
{
Number x,y,tmp,ser;
static Number cof[6]={76.18009172947146,-86.50532032941677,
24.01409824083091,-1.231739572450155,
0.1208650973866179e-2,-0.5395239384953e-5};
int j;
y=x=xx;
tmp=x+5.5;
tmp -= (x+0.5)*log(tmp);
ser=1.000000000190015;
for (j=0;j<=5;j++) ser += cof[j]/++y;
return -tmp+log(2.5066282746310005*ser/x);
}
/*************************************************************
********/
/*Program written by Niraj Nand using the error function written by C. A
Bertulani
in Normal distribution*/
//#define PI 3.141592654
# define v 0.19947114020071633897 //1/(2sqrt(2*PI))
```

```
# define x 0.15915494309189533577 // 1/(2*PI)
# define z 100 //(refine to 5 dp )
// g is the distance and s is  intial value x0
# define g 13
# define s .00001
# define w 2 // Number of stages

/*
Recursive
function receives the parameter k and dk,yk to calculate f.
*/

double RootVal(int k, double d, double y); // calculates the value of the
minimal elements

double Minimum(double val1,double val2){if(val1<=val2){return
val1;}else{return
val2;}}// returns minimum of 2 numbers
double fun(int,int,double ,int,int ,bool );
//
const double inc = 0.001; //PRECISION AMMOUNT
```

```
const double inc2 = 0.00001; //PRECISION AMMOUNT
const double prec = 1/inc;
const int stages = 8;
const int points = 1000 ; //Keep this to be 1/inc
const int factor =4;
//When passing parameter to function . n = your value divid by inc to make it
precise.
// eg. function(3,1) will be passed as function(3,1000)
int ylimits[10];//stores the 3dp values for refining
double minkf2[stages][g*points*z+1];//stores minimum f to 6dp
double dk2[stages][g*points*z+1];//stores minimum d for the 6dp
calculations
void main()
{
//initialize minkf
cout<<"Initializing points ...."<<endl;
for (int i=0; i < stages;i++)
for(int j=0;j<(g*points+1);j++)
minkf2[i][j]= -9999;
for (int k=0; k < stages;k++)
for(int l=0;l<g*points*z+1;l++)
```

```
minkf2[k][l]= -9999;

cout<<"Initialiation

complete"<<endl<<endl<<"Calculating...."<<endl<<endl;

double f=fun(w,g*points,inc ,0,g*points ,true);//f =

printf("\nf(w,g): %.10f \n" ,f);

float d6,d5,d4,d3,d2,d1, y6,y5,y4,y3,y2,y1;

int temp;

//backward calculation for the 3dp results

d6 = g;

y6 = dk2[6][g*points];

d5=d6-y6;

temp = d5*points;

y5=dk2[5][temp];


d4=d5-y5;

temp = d4*points;

y4=dk2[4][temp];

d3=d4-y4;

temp = d3*points;

y3=dk2[3][temp];

d2=d3-y3;
```

```
temp = d2*points;
y2=dk2[2][temp];
d1=d2-y2;
y1=d1;
printf("\nd6: %f y6: %f",d6,y6);
printf("\nd5: %f y5: %f",d5,y5);
printf("\nd4: %f y4: %f",d4,y4);
printf("\nd3: %f y3: %f",d3,y3);
printf("\nd2: %f y2: %f",d2,y2);
printf("\nd1: %f y1: %f",d1,y1);
//setup the limits for the 6dp calculations
temp = y6*points*z;
ylimits[6] = temp;
temp = y5*points*z;
ylimits[5] = temp;
temp = y4*points*z;
ylimits[4] = temp;
temp = y3*points*z;
ylimits[3] = temp;
temp = y2*points*z;
ylimits[2] = temp;
```

```
temp = y1*points*z;

ylimits[1] = temp;

printf("\n\nRefining...\n");

f=fun(w,g*z*points,inc2 ,ylimits[w]- factor*z,ylimits[w]+ factor*z ,false);//

printf("\n\nAccurate values derved after refining\n");

printf("\nf(w,g): %.10f \n" ,f);

//Backward calucation for the 6 dp

d6=g;

y6 = dk2[6][g*points*z];

d5=d6-y6;

temp = d5*points*z;

y5=dk2[5][temp];

d4=d5-y5;

temp = d4*points*z;

y4=dk2[4][temp];

d3=d4-y4;


temp = d3*points*z;

y3=dk2[3][temp];

d2=d3-y3;

temp = d2*points*z;
```

```
y2=dk2[2][temp];
d1=d2-y2;
y1=d1;
printf("\nd6: %f y6: %f",d6,y6);
printf("\nd5: %f y5: %f",d5,y5);
printf("\nd4: %f y4: %f",d4,y4);
printf("\nd3: %f y3: %f",d3,y3);
printf("\nd2: %f y2: %f",d2,y2);
printf("\nd1: %f y1: %f",d1,y1);
getch();
} //end main
double RootVal(int k, double d, double y)//calculate the root value of the
current distribution
{
double rtval;
double calc;
//calc=v*(d-y-s)*exp(-1*pow((d-y-s),2)/2)*geterf((d-s)/sqrt(2))-v*(d-
s)*exp(-1*pow((d-s),2)/2)*geterf((d-s)/sqrt(2))-v*(d-y-s)*exp(-1*pow((d-y-
s),2)/2)*geterf((d-y-s)/sqrt(2)) + v*(d-s)*exp(-1*pow((d-s),2)/2)*geterf((d-y-
s)/sqrt(2)) + w * pow((geterf((d-s)/sqrt(2))- geterf((d-y-s)/sqrt(2))),2) - x*
pow((exp(-1*pow((d-y-s),2)/2) -exp(-1*pow((d-s),2)/2)),2);
```

```
calc=((-1*0.5*exp(2)*(geterf((2-log(d+s))/sqrt(2))-geterf((2-log(d-
y+s))/sqrt(2))))*(0.5*(geterf((log(d+s))/sqrt(2))-geterf((log(d-
y+s))/sqrt(2))))-pow((0.5*exp(0.5)*(geterf((1-log(d+s))/sqrt(2))-geterf((1-
log(d-y+s))/sqrt(2)))),2));
if(calc<0)
{
// cout<<"\nError: Negative Root\n";
// rtval = -1;
}
else
{
calc = sqrt(calc);
}
rtval = calc;
return rtval;
}
//
double fun(int k,int n,double incf,int minYk,int maxYk,bool isFirstRun)//this
functions performs the same actions as "function".
//it only defers in terms of the iterations of the for loop.
```

```
{
assert (k>=1); //Abort if k is negative
double dblRetVal;
double d =n*incf; //d value for the function
double y;
double min;
double val;
double miny;
int col;
if(k==1) //base case
{
y = d;
dblRetVal = RootVal(k,d,y);
}
else
{
for(int i=minYk;i<=maxYk;i++)//iterate over the interval allowed to
calculate the 6dp results.
{
y = i*incf;//this sets to precission of y to 6dp
double root;
```

```
root = RootVal(k,d,y); //calculate the root.
if(root != -1) //if root is valid
{
col =n-i;//get the current d value
if(minkf2[k-1][col]==-9999) {//check if the result has been previously
calculated
if(isFirstRun){
val = root+ fun((k-1),col,incf,0,col,true);//if not, calculate the result
}
else{
val = root+ fun((k-1),col,incf,ylimits[k-1]-
factor*z,ylimits[k-1]+ factor*z,false);//if not, calculate the result
}
}
else
val = root+ minkf2[k-1][col];//if result exists, use it for calculations
}
if (i==minYk)
{
min =val;//base case
}
```

```
else

{
min = Minimum(min,val);//get the minimum if the result and the current
mininmum
}
if(min == val){miny=y;}//get the position of the current minimum
}//end for
dblRetVal = min;
}//end else
//store the f and the d value of the minimum calculated.
col = n;
minkf2[k][col] = dblRetVal;
dk2[k][col]=miny;
return dblRetVal;
}//end function
```

REFERENCES

Aoyoma, H. (1954): A study of the stratified random sampling. *Ann. Ins. Statist. Math.*, **6**, 1-36.

Arthenari, T. S, and Hodge, Yodolah (1981): *Mathematical Programming in Statistics*. John Wiley, New york.

Buhler, W., and Deutler, T. (1975). Optimal Stratification and grouping by dynamic programming. *Metrika,* **22,** 161-175.

Cochran, W. G (1961): Comparison of methods for determining stratum boundaries. *Bull. Int. Statist. Ins.,* **32(2),** 345-358.

Cochran, W. G (1977): *Sampling Techniques*. John Wiley and Sons, New York.

Dalenius, T. (1950): The problem of optimum stratification. *Skandinavisk Aktuarietidskrif.,* **33 ,** 203-213.

Dalenius, T. and Hodges, J. L. (1957): The choice of stratification points. *Skandinavisk Aktuarietidskrift*, 198-203.

Dalenius, T. (1957): *Sampling in Sweden: Contributions to the methods and theories of sample Survey Practice*, Almqvist and Wiksell, Stockholm.

Dalenius, T., and Hodges, J. L., Jr. (1959). Minimum variance stratification. *Jour. Amer. Strat. Assoc.*, **54**, 88-101.

Detlefsen, R. E. and Veum, C. S. (1991). Design Issues for the Retail Trade Sample Surveys of the U. S. Bureau of the Census. Proceedings of the Survey Research Methods Section, ASA, PP.214-219.

Dalenius, T., and Gurney, M. (1951). The problem of optimum stratification. II. *Skand. Akt.*, **34,** 133-148.

Durbin, J (1959). A note on the application of Quenouille's method of bias reduction to the estimation of ratios. *Biometrika*, **46** , 477-480.

Ekman, G. (1959): An approximation useful in univariate stratification. *Ann. Math. Statist.*, **30**, 219-229.

Hansen, M. H., Hurwitz, W. N. and Madow, W. G. (1953): *Sample survey methods and theory.* John Wiley, New York.

Hess, I., Sethi , V. K., and Balakrishnan, T. R. (1966): Stratification: A Practical investigation. *J. of Amer. Statist. Assoc.*, **61**, 74-90.

Hidiroglou, M. A., and Srinath, K. P. (1993). Problems Associated with Designing Subannual Bussiness Surveys. *Journal of Bussiness and Economic Statistics*, **11**, 397-405.

Khan, E. A., Khan, M. G. M. and Ahsan, M. J. (2002): Optimum stratification: A mathematical programming approach. *Calcutta Statistical Association Bulletin*, **52** , 323-333.

Khan, M. G. M., Najmussehar and Ahsan, M. J. (2005):Optimum Stratification for Exponential Study variable under Neyman Allocation. *Journal of Indian Society of Agricultural Statistics*, **59(2),** 146-150.

Kozak, M. (2004). Optimal Stratification using random search method in agricultural surveys. *Statistics in Transition*, **6(5),** 797-806.

Lavallee, P. (1987): Some contributions to optimal Stratification, Master Thesis, Carleton University, Ottawa Canada.

Lavallee, P., and Hidiroglou, M. (1988). On the stratification of skewed populations. *Survey Methodology*, **14,** 33-43.

Lednicki, B., and Wieczorkowski, R. (2003). Optimal Stratification and sample allocation between subpopulations and strata. *Statistics in Transition*, **6**, 287-306.

Mahalanobis, P. C. (1952): Some aspects of the design of Sample surveys. *Sankhya*, **12,** 1-7.

Murti, M. N. (1967): Sampling Theory and Methods .Statistical Publishing Society, Calcutta.

Nand, N. et al (2008): Unpublished thesis of A. H. Ansari (2008).

Nelder, J. A., and Mead, R. (1965)A simplex method for function minimization. Computer Journal, **7,** 308-313.

Nicolini, G. (2001): A method to define strata boundaries. Working paper 01 -2001-marzo, Departimento di Economia Politica e Aziendale, Universita degli Studi di Milano.

Niemiro, W. (1999). Konstrukcja optymalnej stratyfikacja method Poszukiwan losowych. (optimal stratification using Random Search Method).Wiadomosci Statystyczne,10, 1-9.

Rivest, L. P. (2002): A Generalization of Lavallee and Hidiroglou Algorithm for Stratification in Business Survey, *Techniques d'enquete*, 28,207-214.

Sethi, V. K (1963): A note on optimum stratification for estimating the population means. *Aus. J. Statist.*, **5**, 20-33.

Sweet, E. M. and Sigman, R. S. (1995a): Evaluation of model – assisted procedures for Stratifying skewed populations using auxiliary data, Proceedings of the survey Research Methods Section, ASA, 491-496.

Sweet, E. M. and Sigman, R. S. (1995b): User guide for the Generalized SAS Univariate Stratification Program, ESM Report Series, ESM -9504, U. S. Bureau of the Census.

Unnithan, V. K. G. and Nair, N. U. (1995): Minimum variance stratification, Communications in Statistics –Simulation and Computation, 24(1), 275-284.

Unnithan, V. K. G. (1978):"The minimum variance boundary points of Stratification". Sankhya, 40, C, 60-72.